■ **Web** I plug-in con cui crei siti WordPress da professionista

L'unica guida libera al mondo dell'Open Source

NEL WEB DVD

le istruzioni a pagina 4

- Zorin OS 16.2 Core
- Peppermint OS 11
- Nitrux 2022.11.02
- Bitfighter 0.22
- HomeBank 5.5.8
- Speed Dreams 2.2.3
- Vup 0.12.0
- Whatsie 4.10.2
- *e molto altro...*

INTELLIGENZA ARTIFICIALE

L'IA ti cambia la vita quando crei e programmi!

- Progetta il tuo motore per capire i sentimenti nei testi
- Scopri i sistemi cloud per automatizzare programmazione e altro
- Ecco come fare doppiaggi ed estrarre strumenti dalle canzoni

8,8 GB DI SISTEMI OPERATIVI E APP!

Zorin OS è l'alternativa perfetta a Windows e macOS!

Questa è la prima copertina creata in Italia con l'Intelligenza Artificiale!

NUOVO! 2 DVD DA SCARICARE CON IL CODICE ALL'INTERNO IN ESCLUSIVA PER TE

Linux per grandi TV
Plasma Bigscreen rivoluziona il modo in cui vedi i contenuti sul televisore!

TARIFFA R.O.C. - POSTE ITALIANE SPA - ABB. POST. - D.L. 353/2003 (CONV. IN L. 27/02/2004 N. 46) ART. 1, COMMA 1, S/NA - DISTRIBUTORE: PRESS-DI, SEGRATE

Bimestrale - N.216 - 6,90 €

P.I. 09-12-2022 Dicembre-Gennaio

Maker Lab
L'assistente è Open Source
» Costruisci facilmente la tua Alexa personale con la Pi!

Coding
Programma il tuo Excel
» Con F# crei il tuo foglio di calcolo con tutti i suoi strumenti

C'è una prima volta per tutto

Saper lavorare con strumenti di machine learning e Intelligenza Artificiale farà la differenza fra chi nel prossimo futuro avrà una carriera professionale nel mondo IT molto remunerativa e chi... un po' meno. A questo scopo abbiamo pensato un articolo di copertina tutto dedicato a questo mondo con anche un tutorial per farvi mettere le mani "in pasta". Ma siccome ci piace rompere barriere, abbiamo pensato di realizzare la prima copertina creata in Italia usando un sistema di Intelligenza Artificiale. Abbiamo detto a DALL·E 2, di cui parliamo nell'articolo, di crearci una fotografia sul concetto di intelligenza artificiale e creatività e, dopo aver scelto la versione migliore che aveva prodotto, abbiamo sfruttato le sue capacità di ampliamento del background per renderla sufficientemente grande da stare nella nostra copertina. Ci auguriamo che i nostri sforzi vi diano l'ispirazione per creare tantissimi progetti usando gli strumenti che vi presentiamo! C'è poi naturalmente molto di più, con il nuovo ambiente per grandi schermi, la panoramica sui plug-in di WordPress per creare siti professionali velocemente e tutorial per spiegarvi come programmare un videogioco o addirittura creare un programma in stile Excel fatto con le vostre mani. Buona lettura!

La redazione di Linux Pro

Da non perdere su questo numero...

NOI RISPETTIAMO L'AMBIENTE!

Linux Pro è stato stampato su carta certificata PEFC, proveniente da piantumazioni a riforestazione programmata e perciò gestite in maniera sostenibile

CONTATTI

Domande alla redazione: **redazione@linuxpro.it**
Abbonamenti e arretrati: **abbonamenti@sprea.it**
Problemi con il DVD: **aiutocd@sprea.it**
Sito Web: **www.linuxpro.it**
Oppure inviate le vostre lettere a:
***Linux Pro*, Sprea S.p.A.,**
Via Torino 51, 20063 Cernusco S/N
Telefono: 02.92432.1

LINUX PRO 216

Sommario

LINUX PRO

Benvenuti nel duecentosedicesimo numero di Linux Pro, la guida definitiva a Linux e al mondo Open Source

Cover story

INTELLIGENZA ARTIFICIALE

Sommario

IL PROSSIMO NUMERO IN EDICOLA DAL 10 FEBBRAIO

IL DVD IN BREVE

DVD A

DISTRIBUZIONI
- Zorin OS 16.2 Core
- Peppermint OS 11

DVD B

DISTRIBUZIONI
- Nitrux 2022.11.02

RIVISTA
- Bitfighter 0.22
- HomeBank 5.5.8
- Vup 0.12.0
- WhatSie 4.10.2
- AESCrypt GUI 3.11
- jExifToolGUI 2.0.1

E altro ancora!

Guida DVD

Ogni mese Linux Pro vi offre i programmi e le distribuzioni più recenti su DVD

Sul DVD di questo mese...

Distro

Zorin 16.2

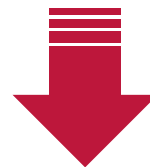
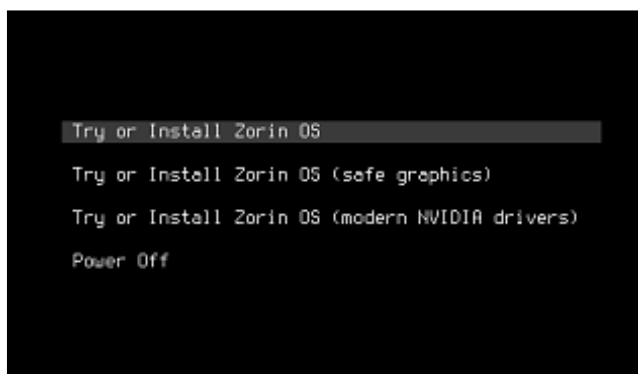
Dopo circa cinque mesi dalla precedente versione, ecco disponibile l'edizione **16.2** di Zorin che va a migliorare ulteriormente la già ottima esperienza utente offerta da questo sistema operativo basato su **Ubuntu 20.04 LTS, Focal Fossa**. A installazione finita vi troverete di fronte a un desktop completamente vuoto, fatto salvo per la barra in basso. Qui però c'è tutto ciò che vi serve, a cominciare dal menu principale che si apre facendo click sul **logo** di **Zorin** in basso a sinistra. Di seguito ci sono i pulsanti che aprono nell'ordine la schermata di ricerca, **Firefox**, la **cartella Home** e lo strumento **Software**, che serve per installare ulteriori applicazioni. A destra invece ci sono due menu. Il primo è per la gestione del collegamento a Internet e delle sessioni, mentre il secondo è il calendario. Per quanto riguarda il menu principale è impossibile non notare una certa somiglianza con quello di **Windows**. A sinistra abbiamo infatti le cartelle tematiche, mentre a destra c'è l'accesso diretto alle **directory** principali, alle impostazioni e al vostro profilo utente. Quindi chi proviene dal sistema operativo di **Microsoft** si sentirà quasi a casa propria.

Le novità

Ad accrescere questa sensazione, in **Strumenti di sistema** di quest'ultima versione di Zorin troverete il **Supporto dell'app di Windows** che, tradotto in parole povere, è **Wine**, il famoso "non emulatore". Grazie a esso potrete installare numerosissimi programmi compatibili con il sistema operativo di Microsoft, con prestazioni di poco inferiori a quelle originali. Non sarà però necessario installare **Office** poiché è già disponibile la versione **7.4** della **suite** per ufficio **LibreOffice**, completa di tutte le applicazioni, fatta eccezione per **Base**, che può essere comunque installata gratuitamente tramite **Software**. Ci sono novità anche sul fronte **Zorin Connect**, lo strumento che permette di integrare alcune funzionalità del vostro **smartphone**. Ora è stata attivata la possibilità di visualizzare lo stato della batteria sul desktop. Inoltre, si possono usare i controlli multimediali da remoto.

La dotazione software

Di sicuro Zorin non è avaro di applicazioni fin dall'installazione. Per esempio avrete **Mappe** e **Meteo** nella sezione **Accessori**. Per quanto riguarda i file multimediali, in dotazione ci sono **Rhythmbox** e **Video**, noto anche come **Totem**, oltre a **Cheese** per la gestione della Webcam. Se invece siete interessati alla fotografia, la presenza di **GIMP** vi farà sicuramente molto piacere. Oltre al già menzionato LibreOffice, nella sezione **Ufficio** avrete disponibile un comodo strumento per la gestione dei vostri contatti. Anche la sezione **Utilità** risponde bene alla maggior parte delle esigenze degli utenti, con applicazioni come **Password e chiavi** (per non dover ricordare tutte le vostre credenziali) e lo strumento per catturare le schermate.

SCARICA SUBITO I DUE DVD COMPLETI INSERENDO IL CODICE AIDALLE

Come si installa e si configura Zorin 16.2

1 Avvio della procedura

La prima schermata che viene visualizzata vi permette di scegliere l'installazione di **Zorin**. Lasciate selezionata la prima opzione e premete **INVIO**. Dopo qualche secondo appare la schermata di benvenuto. A sinistra selezionate **Italiano** e a destra fate click su **Installa Zorin OS**.

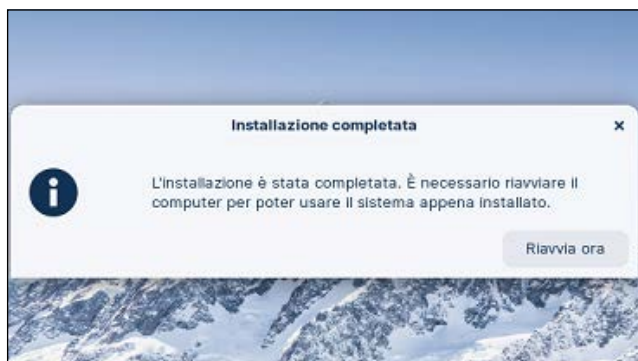
2 Tastiera e censimento

Come vedrete nella schermata che segue, la tastiera italiana è già selezionata correttamente quindi, se non ne avete una diversa, fate click su **Avanti**. In **Aggiornamenti e altro software**, se non volete partecipare al censimento, selezionate la terza opzione che vedete e premete su **Avanti**.

3 Tipo di installazione

Ora potete scegliere il tipo di installazione. Se non volete modificare le partizioni standard, lasciate selezionato **Cancella il disco e installa Zorin OS** e fate click su **Installa**. A questo punto appare una **finestra pop-up** che vi chiede di confermare la formattazione delle partizioni indicate. Premete su **Avanti** per farlo.

4 Localizzazione e profilo

La schermata **Località** mostra **Rome** come fuso orario predefinito. Confermatelo facendo click su **Avanti**. In **Informazioni personali**, compilate i campi **Il vostro nome**, **Scegliere una password** e **Confermare la password**. Selezionate ora **Accedere automaticamente** e fate click su **Avanti**.

5 Installazione e riavvio

A questo punto inizia l'installazione vera e propria di **Zorin OS** sul vostro computer, in base ai parametri scelti finora. Aspettate che l'operazione finisca e poi fate click su **Riavvia ora**. Poi, quando richiesto, premete il tasto **INVIO** per scollegare automaticamente il supporto con il file di installazione.

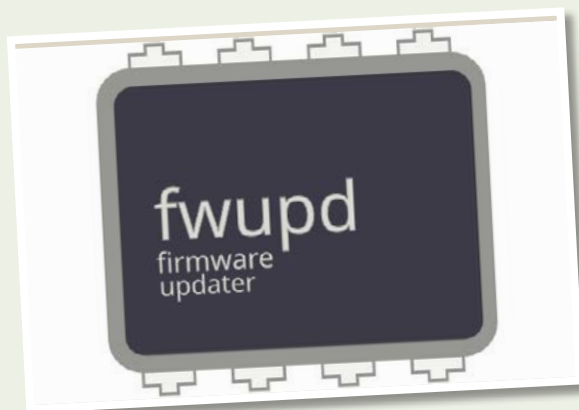
6 Il primo avvio

Fate click su **No grazie** nella finestra **Benvenuti in Zorin OS 16.2** se volete saltare la visita guidata alle funzioni del sistema operativo. Subito dopo vedrete che si è aperta la finestra **Aggiornamenti software**. Premete su **Installa ora** e, finito l'aggiornamento, riavviate il sistema operativo. LXP

Newsdesk

Il nuovo firmware Linux è ora disponibile

La versione 1.8.7 è in grado di supportare un maggior numero di GPU Intel Discrete e di laptop Star Labs

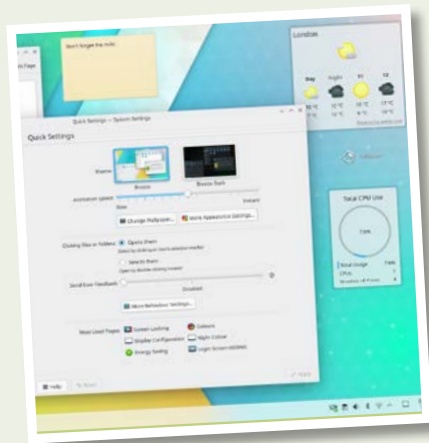
Nelle scorse settimane è stata rilasciata la nuova versione dell'**utility** di aggiornamento del **firmware** per tutte le distribuzioni **GNU/Linux**. Con questa uscita si è arrivati all'edizione **1.8.7** che, oltre alla solita serie di **bug** corretti, introduce caratteristiche nuove e interessanti per tutti gli utenti, che si traducono in un supporto per un ventaglio sempre maggiore di hardware. Tra i beneficiari di questo nuovo firmware troviamo i **laptop** marcati **Star Lab** tra cui, a quanto sembra, ci sarà anche il nuovissimo **StarFighter**, che sarà il primo computer portatile della casa ad avere uno schermo con una risoluzione **Ultra HD**, cioè a **4K**. Anche le unità di elaborazione grafica discreta **Iris Xe MAX** di **Intel**, dedicate ai laptop, verranno supportate da questa nuova versione del firmware, sebbene solo in modo sperimentale per il momento. Naturalmente il suo campo d'azione non si limita a questi hardware specifici, poiché sono moltissimi i nuovi dispositivi supportati, come i **touchpad** di **Elan** e vari sistemi per il riconoscimento delle impronte digitali. Inoltre viene introdotto il supporto per il **Mini Hub Thunderbolt 4** di **Anker** e per il **QSI Godzilla Creek Reference Hub**.
Sul fronte software abbiamo il supporto per il formato di compressione **xz** per i **metadati** che, secondo le stime, dovrebbe ridurre del **25%** l'occupazione della banda per lo scaricamento dei file firmware. Viene inoltre introdotta la capacità di misurare l'integrità del sistema quando vengono installati gli aggiornamenti dell'**UEFI**. Per quel che riguarda la correzione dei bug, vale la pena mettere in evidenza la nuova capacità di analizzare metadati di dimensioni superiori a **1 MB**, il controllo di correttezza dei permessi dei file di configurazione dei **plug-in** integrati e l'aggiunta della richiesta di reinserimento per i dispositivi **Analogix**. Per poter aggiornare il vostro firmware, o anche solo per ottenere ulteriori informazioni sulla sua ultima versione, potete collegarvi all'indirizzo **https://bit.ly/3Txi2Za**. Alla pagina **https://bit.ly/3TxghLF** trovate invece la **wiki** del progetto, che offre informazioni per distribuzioni specifiche e altri approfondimenti tecnici.

Moltissime sono le novità che porta con sé questo nuovo firmware soprattutto per quanto riguarda i nuovi hardware supportati

PROVE DI SUPPORTO

Il nuovo firmware supporterà in via sperimentale le GPU Iris Xe Max di Intel progettate per i laptop

KDE Plasma aumenta la compatibilità con Wayland

Vediamo quali miglioramenti riserva la versione 5.26.3 del noto ambiente desktop

All'inizio di novembre e a sole due settimane dal precedente, **KDE Project** ha rilasciato il terzo aggiornamento della versione **5.26** del suo famoso e apprezzato ambiente desktop. Una delle prime cose a cui è stata messa mano è stata la correzione di certi problemi riscontrati e segnalati dagli utenti, per esempio con la nota applicazione videoludica **Steam**, ma non solo. Il difetto riguardava il ridimensionamento automatico delle finestre dei programmi che non avveniva correttamente nelle sessioni **Wayland**. Un altro problema risolto nella stessa sessione riguarda il trascinamento di elementi all'interno di una finestra di **Firefox**. Infatti capitava che il cursore vi restasse bloccato durante l'operazione. È stato anche sistemato un fastidioso **bug** che provocava il **crash** dell'ambiente desktop quando veniva usata l'applicazione **Plasma Vaults**. Invece tra gli aggiornamenti che porta con sé questa nuova versione abbiamo quelli relativi al **Plasma NetworkManager** per permettergli di riconoscere correttamente la libreria **libreswan 4.9**. Per chi usa questo ambiente desktop su dispositivi mobili è stato inoltre migliorato il supporto per i **tablet**. Le note di rilascio sono all'indirizzo **https://tinyurl.com/bdfsjdjs**.

I miglioramenti apportati alla versione 5.26 di KDE Plasma riguardano anche i dispositivi portatili

L'Aragosta Lunare

Svelato il nome in codice del nuovo Ubuntu

Come da tradizione ampiamente consolidata in casa **Canonical**, la primissima informazione certa che si ha sul sistema operativo che succede a quello appena rilasciato è il nome in codice. Già si sapeva che sarebbe cominciato con la lettera **L**, che nell'alfabeto inglese segue la **K** appena usata per il recente **Kinetic Kudu** (**Ubuntu 22.10**). Questa volta per trovare l'animale simbolo della versione **23.04** si è lasciata la terraferma e ci si è gettati nel mare pescando **Lunar Lobster**, che significa aragosta lunare. Inutile cercare indizi in questo nome perché ormai Canonical li sceglie perché li trova simpatici. La data di rilascio è stata attualmente fissata per il **27 aprile 2023** e sono anche già state stabilite le tappe precedenti. Per esempio la **Feature Freeze** sarà il **23 febbraio**, mentre l'interfaccia definitiva sarà stabilita il **16 marzo** e la **versione beta** sarà disponibile dal **30** dello stesso mese. Per quanto riguarda invece le novità che dovrebbero esserci, le voci che circolano sono parecchie. Per il momento non è chiaro se verrà adottata come **kernel** la versione **6.1** o la **6.2** di Linux. Inoltre è possibile che venga introdotto il nuovo **Ubuntu Software**, già da tempo in lavorazione, così come il nuovo programma di installazione. Si parla anche dell'ambiente grafico **GNOME 44** e della presenza nei **repository** del nuovo sistema operativo di librerie software più aggiornate, come **PHP 8.2** e **Python 3.11**.

SIPEED CON UBUNTU

Lo scorso **25 ottobre Canonical** ha annunciato ufficialmente che il suo sistema operativo verrà ora supportato dai **single-board computer LicheeRV D1 RISC-V** marcati **Sipeed**. Si tratta quindi dell'ennesima operazione riuscita nell'ambito del progetto di espansione nel mondo **RISC-V**. La scheda in questione, che ha un costo molto conveniente perché si aggira sui **16 $**, viene prodotta dalla cinese Sipeed che punta tutto sull'hardware Open Source e sulle applicazioni **AIoT/TinyML**. Quindi non c'è da stupirsi che Canonical possa rappresentare un partner ideale per lo sviluppo di progetti futuri. Tornando alla scheda in questione, è stata pensata proprio per gli sviluppatori TinyML ed è dotata di una porta **USB-C OTG** e una **HDMI**. Inoltre i suoi **43,2 x 25 mm** hanno a bordo **512 MB** di **RAM DD3** a **792 MHz** e un socket per schede **MicroSD**. Il processore è un **Allwinner D1 XuanTie C906 single-core** da **1 GHz** a **64 bit** con un acceleratore grafico **G2D 2D**.

MOTORI RUGGENTI!

È passato più di un anno dalla pubblicazione di **SuperTuxKart 1.3** ed ecco che finalmente arriva rombando la versione successiva, la **1.4**. Essendo una **major release**, ha diverse novità sotto al cofano, a cominciare dalla disposizione di partenza dei **kart** per rendere le gare più avvincenti. Anche alcuni percorsi, come **Battle Island**, sono stati migliorati o hanno visto corretti alcuni difetti. Invece tra le novità vere e proprie abbiamo una nuova modalità di giro di prova, un veicolo aggiuntivo e svariate **texture** e animazioni. Dal punto di vista tecnico è stato invece introdotto il nuovo **renderizzatore Vulkan** che è ancora in **versione beta** ma che già garantisce risultati di ottimo livello. Per poter scaricare gratuitamente il file di installazione del videogioco completo, basta collegarsi all'indirizzo **https://supertuxkart.net/Download** e fare click sul sistema operativo, in questo caso **Linux**.

LibreOffice 7.4 ancora più stabile

Moltissimi bug risolti nella nuova versione della suite per ufficio

The Document Foundation sta spingendo perché gli utenti passino alla versione 7.4

Dal rilascio dello scorso agosto, la più recente **release** di **LibreOffice** ha già compiuto i primi passi avanti e attualmente è giunta alla versione **7.4.2**, che segue di solo un mese la precedente. Il suo principale punto di forza è la risoluzione di ben **80 bug**, che rende la nuova suite per ufficio targata **The Document Foundation** finalmente appetibile. Infatti, fino a questo momento, la stessa fondazione suggeriva ai propri utenti di continuare ad affidarsi alla versione **7.3.6** di LibreOffice, considerata più stabile e affidabile. Con l'arrivo di questa nuova edizione la situazione cambia in modo radicale e la politica aziendale ora punta sulla 7.4, disponibile all'indirizzo **https://www.libreoffice.org/download/download-libreoffice/** in formato **DEB** o **RPM**, in base alla distribuzione Linux che avete. Il supporto per questa nuova versione della suite per ufficio durerà fino a **giugno 2023** mentre è stata annunciata una nuova **point release** per fine novembre che dovrebbe essere disponibile al momento dell'uscita della nostra rivista in edicola e garantire ancor maggiore affidabilità.

Messaggistica nativa in Firefox

Finalmente anche gli utenti di Ubuntu ce l'avranno

Succede a volte che lo sviluppo di un'applicazione non sia omogeneo in tutte le sue versioni e che alcuni utenti possano sfruttarne una caratteristica non ancora disponibile per altri. È il caso della messaggistica nativa di **Firefox** che, nel sistema operativo di **Canonical**, non era ancora presente. La causa principale era la versione in formato **Snap** del browser di **Mozilla** adottata da **Ubuntu**. In realtà, il problema era già stato risolto fin dallo scorso luglio, tuttavia questa caratteristica era presente solo nelle versioni **beta** del programma di navigazione. Invece ora gli utenti di Ubuntu la troveranno perfettamente integrata e soprattutto stabile. Così ha annunciato nel forum **Snapcraft** uno dei tecnici informatici di Canonical che si è occupato del problema, **Olivier Tilloy**. Ora gli **add-on** di **Firefox Snap** potranno scambiare messaggi con le applicazioni native del sistema operativo creato da **Mark Shuttleworth**.

Disney+ non va d'accordo con Linux

Per fortuna c'è chi ha risolto questo problema

La buona notizia per gli utenti **Linux** è che possono finalmente usufruire dei contenuti offerti da **Disney+** grazie a **gnif**, uno **YouTuber** esperto di informatica. Basterà seguire le istruzioni presenti nel video che trovate all'indirizzo **https://youtu.be/IS4BCzwSA6M**. In pratica si tratta di far credere al noto canale di **streaming** a pagamento che invece di Linux state usando **Windows**, modificando alcuni parametri.

Per il momento gli utenti Linux dovranno agire di persona per poter vedere Disney+

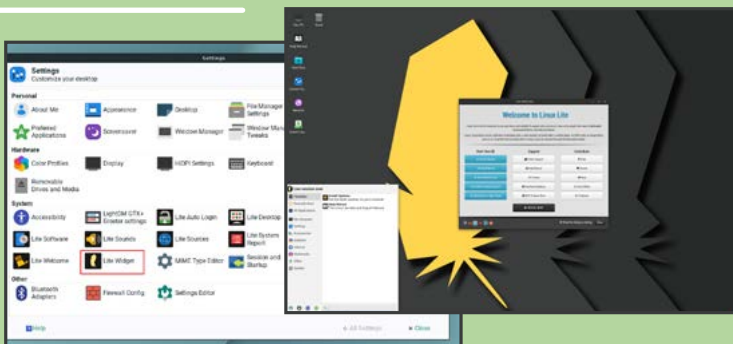
Mondo distribuzioni

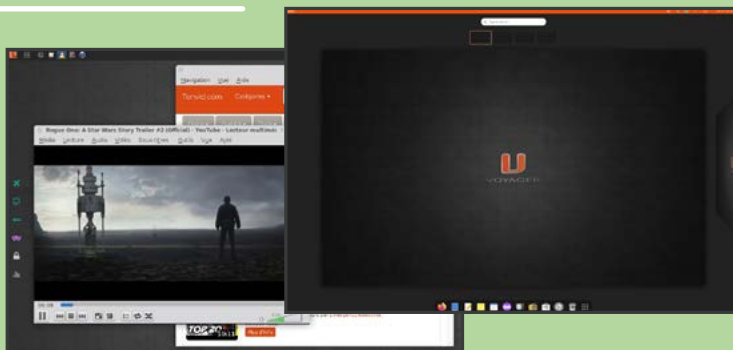
Le nuove uscite e i progetti da scoprire

NITRUX

Nitrux è una distribuzione **GNU/Linux** basata su **Debian Unstable**, o **Sid**, a cui sono stati aggiunti pacchetti dei repository di **Ubuntu LTS**. Il suo ambiente desktop è basato su una versione potenziata di **KDE Plasma** che si chiama **NX**. Inoltre, questo progetto si concentra sull'uso di applicazioni portatili in formato **AppImage**. La nuova **release** (**20221101**) che può essere scaricata all'indirizzo **https://tinyurl.com/yrh7keu7** ha tra le sue più importanti novità la presenza dei **driver** proprietari **NVIDIA**.

LINUX LITE

Linux Lite si contraddistingue per essere una distribuzione adatta ai principianti. Basata su **Ubuntu LTS** e con l'ambiente desktop **Xfce**, è attualmente alla sua versione **6.2**, ottenibile all'indirizzo **www.linuxliteos.com/download.php**. Gli sforzi degli sviluppatori si sono concentrati sul miglioramento della gestione degli aggiornamenti e sono state adottate alcune nuove applicazioni, come **Shotcut** al posto di **Openshot**.

VOYAGER LIVE

Voyager Live 22.10 è una distribuzione basata su **Xubuntu** con ambiente desktop **Xfce** o **GNOME** e ottenibile all'indirizzo **https://tinyurl.com/4a9dedmt**. Tra le sue caratteristiche c'è l'**Avant Window Navigator**, o **AWN**, che è una barra di navigazione in stile **dock**. L'attuale versione è stata completamente riprogettata per permettere la coesistenza dei due ambienti desktop.

PEPPERMINT OS

Peppermint OS (**https://peppermintos.com/guide/downloading**) è una distribuzione **GNU/Linux** attualmente basata su **Debian**, mentre in precedenza era fondata su **Lubuntu**. Poco avida di risorse e facile da usare, da quest'anno ha abbandonato l'ambiente desktop ibrido **LXDE/Xfce** per passare completamente a quest'ultimo. Grazie al suo browser dedicato, integra senza problemi applicazioni basate sul Web e sul **cloud**. Inoltre si aggiorna automaticamente. LXP

INTELLIGENZA ARTIFICIALE

Le sempre nuove applicazioni dell'Intelligenza Artificiale possono offrire infiniti strumenti per il vostro lavoro e anche per la vostra creatività

Per fare una panoramica del mondo dell'Intelligenza Artificiale non basterebbe un'intera rivista.
Oggi tutto, dagli scenari più quotidiani come comprare un paio di scarpe sportive perfette per il nostro piede a quelli più creativi come realizzare quadri ispirati allo stile del nostro pittore preferito, può avere una marcia in più grazie alle infinite applicazioni di questa disciplina. Anche opportunità lavorative che una volta avrebbero richiesto molto tempo, come scrivere una presentazione in un inglese senza sbavature e naturale e anche inserire una traccia audio che ne legge il testo con un accento da madrelingua, sono a portata di mano (ed economicamente gestibili!) grazie all'Intelligenza Artificiale. Persino il suo stesso sviluppo è facilitato... da strumenti basati su **IA** e **apprendimento automatico** (o **ML** da **Machine Learning**) e **Python** è un ottimo linguaggio per cimentarsi con le nuove frontiere dell'informatica. Gli strumenti Open Source a vostra disposizione sono numerosi ed è sicuramente un campo che può dare grandi soddisfazioni. Come vedremo nelle prossime pagine, però, ci sono anche delle questioni etiche che gli operatori di questo campo si stanno già attrezzando per affrontare.

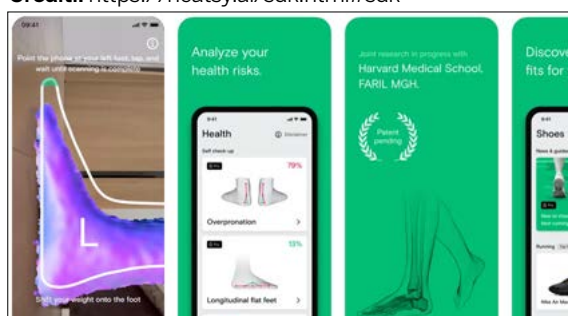
Cosa si fa oggi con l'Intelligenza Artificiale

Prima di mettere le mani sul codice, diamo un'occhiata insieme a quello che le aziende stanno creando oggi in questo settore

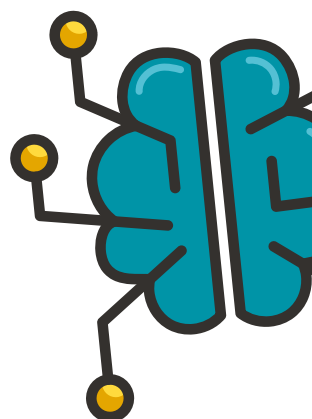
Se pensiamo al termine Intelligenza Artificiale siamo portati a immaginare automobili a conduzione automatica e computer che giocano a scacchi, ma ormai i campi applicativi di questa disciplina sono infiniti, come dimostra il numero di **startup** che continuamente nascono nel settore.

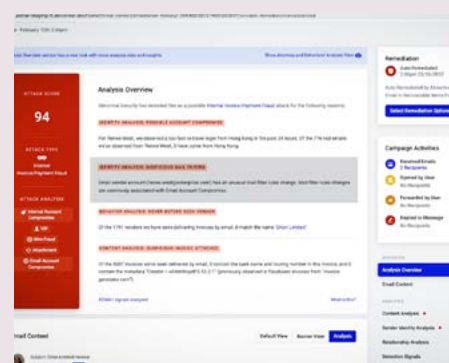
Dalle scarpe alle tecniche di scrittura

Anche un'operazione banale come comprare un paio di scarpe può beneficiare dell'IA. Con **Neatsy** (**neatsy.ai**), infatti, basta un **iPhone** per fare una scansione tridimensionale del piede e non solo trovare le scarpe perfette senza impazzire con i numeri, ma anche individuare possibili problemi ortopedici presenti e futuri. Per gli utenti significa maggior facilità negli acquisti online e per i negozi un minor rischio di reso. Passando dai piedi alla testa, se il vostro problema è dover comunicare in inglese con un'efficacia professionale, l'Intelligenza Artificiale vi aiuta in **Linguix** (**linguix.com**). Mentre scrivete il vostro testo, potete vedere suggerimenti sulla grammatica, la sinteticità, le frasi efficaci, i **cliché** da evitare, il linguaggio potenzialmente offensivo e molto altro ancora. Un salvavita, per esempio, per team di marketing che lavorano con l'estero.

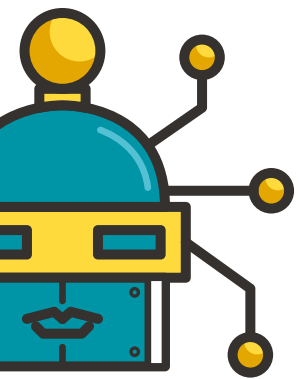
Crediti: https://neatsy.ai/sdk.html#sdk

L'app **Neatsy** utilizza algoritmi generati dall'**IA** per trovare la misura perfetta per le scarpe e identificare potenziali disturbi podologici attraverso la scansione dei piedi tramite **iPhone**

Meta-Intelligenza Artificiale

Usare l'Intelligenza Artificiale per creare l'Intelligenza Artificiale è già realtà con **H2O.ai** che mira alla "democratizzazione" dell'IA. Nata da un gruppo di persone dalla mentalità affine nella comunità Open Source, offre la piattaforma **H2O AI Cloud** che "consente alle aziende, agli enti governativi, alle organizzazioni **non profit** e alle istituzioni accademiche di produrre, operare e innovare con l'Intelligenza Artificiale per accelerare l'innovazione responsabile e spingere i confini di ciò che è possibile con essa".
Per saperne di più sulla piattaforma **H2O Open Source**, visitate il sito **https://bit.ly/3NYW0gQ**.

» PROFILAZIONE COMPORTAMENTALE CONTRO I PIRATI

Uno dei problemi più gravi in ambito informatico sono gli attacchi facilitati dall'errore umano: con tecniche di **social engineering** e truffe elaborate e mirate, come impersonare persone fidate, i pirati possono rubare i vostri preziosi dati o rovinare la vostra azienda. La tecnologia può fare poco contro l'errore umano, ma l'Intelligenza Artificiale arriva anche qui. **Abnormal Security** (**abnormalsecurity.com**) è un'azienda che si occupa di sicurezza delle **email** e che sfrutta l'Intelligenza Artificiale per proteggere le aziende da attacchi mirati. Crea infatti un **profilo comportamentale** degli utenti, dell'organizzazione e dei fornitori. Ogni messaggio viene analizzato utilizzando decine di migliaia di segnali relativi all'identità, al comportamento e al contenuto per identificare le anomalie e rilevare i comportamenti malevoli, compresi gli attacchi mai visti prima e privi di indicatori tradizionali di compromissione.

L'Intelligenza Artificiale applicata alla sicurezza permette di prevenire attacchi riconoscendo comportamenti anomali

Intelligenza Arte-ificiale

DALL·E 2 è forse il sistema di IA più discusso del momento: ecco cos'è, come funziona e perché sta rivoluzionando il mondo della creatività

DALL-E 2 è un nuovo sistema di Intelligenza Artificiale in grado di creare immagini e arte realistici a partire da una descrizione in linguaggio naturale. Può combinare concetti, attributi e stili. Il suo nome deriva dalla fusione di quelli dell'artista Salvador Dalì e del robot WALL-E dell'omonimo film della Pixar.
Ed è un nome molto azzeccato, perché fa subito pensare all'impatto emotivo che può avere l'Intelligenza Artificiale. Al suo lancio nel gennaio **2021**, DALL-E ha lasciato tutti a bocca aperta. Un anno dopo, la sua creatrice **OpenAI** ha introdotto un nuovo sistema, **DALL-E 2**, che genera immagini più realistiche e precise con una risoluzione quattro volte superiore. Ma come è nato tutto questo?

In principio era il Verbo

Nel 2020, OpenAI ha rilasciato **GPT-3** (**Generative Pre-trained Transformer 3**), un **modello linguistico autoregressivo** che utilizza il **deep learning** per generare testi. Dato un testo iniziale come **prompt**, per esempio, ne genera uno che continua la narrazione. Rappresenta la terza generazione della serie **GPT** di OpenAI, un laboratorio che conduce ricerche con l'obiettivo dichiarato di promuovere e sviluppare forme di Intelligenza Artificiale a beneficio di tutta l'umanità. L'organizzazione è stata fondata alla fine del **2015** da un gruppo di imprenditori, tra cui **Elon Musk**, che si sono impegnati collettivamente per un miliardo di dollari. L'architettura di GPT-3 (**https://arxiv.org/abs/2005.14165**) si basa su una rete di **transformer** modificati con dimensioni del contesto di ben **2048 token** (vedi seguito) e con **175 miliardi** di parametri. Un transformer è un modello di **deep learning** che adotta il meccanismo dell'**autoattenzione**, ponderando in modo

DALL·E 2 è in grado di creare immagini concettualmente stupefacenti. Per esempio, qui sopra potete vedere come sia in grado di espandere un dipinto (il famosissimo Ragazza col turbante di Jan Vermeer) creando un ambiente circostante mentre a destra ne vedete una reinterpretazione

differenziato il significato di ciascuna parte dei dati in ingresso. GPT-3 sfrutta il metodo di apprendimento chiamato **pre-allenamento generativo** che mira a prevedere quale sarà il prossimo token. Il sistema è in grado di scrivere testi originali indistinguibili da quelli di un essere umano. DALL-E è una versione a **12 miliardi** di parametri di GPT-3 addestrata a generare immagini da descrizioni testuali, utilizzando un **dataset** di coppie testo-immagine.

Dalle parole alle immagini

DALL-E riceve sia il testo sia l'immagine come un singolo flusso di dati contenente fino a **1.280** token, che possono essere descritti come qualsiasi simbolo di un vocabolario discreto. Per esempio, per gli italiani ogni lettera romana è un token di un alfabeto di **26** lettere. Il vocabolario di DALL-E contiene sia i token per i concetti testuali sia quelli delle immagini. In particolare, ogni didascalia di un'immagine è rappresentata utilizzando un massimo di **256** token da un vocabolario di **16.384**, mentre le immagini sono rappresentate con **1.024** token di vocabolario di **8.192**. Nel processo di generazione delle immagini, un **encoder** prende il testo e genera **embedding testuali**. Questi ultimi, semplificando, rappresentano la conversione del testo in forma vettoriale in modo che la macchina possa sviluppare connessioni tra i vettori e le parole e comprendere il linguaggio. Questi embedding fungono da input per un modello chiamato **prior**, che genera gli embedding corrispondenti delle immagini, da cui infine un modello di decodifica genera le immagini vere e proprie. Per selezionare la migliore, DALL-E utilizza **CLIP** (**Contrastive Language-Image Pre-training**), un modello separato che è stato addestrato su **400 milioni** di coppie di immagini e didascalie testuali prese da Internet (**https://openai.com/blog/clip/**). In DALL-E 2, il decodificatore è un altro modello creato da OpenAI, chiamato **GLIDE** (**Guided Language to Image Diffusion for Generation and Editing**).

Ecco a voi "Tux il pinguino di Linux che programma del codice su una spiaggia deserta"

Il software (**https://arxiv.org/abs/2112.10741**) è un **modello di diffusione** modificato che, a differenza di quanto avviene normalmente, include informazioni testuali. Una serie di tecnologie d'avanguardia è quindi confluita nella creazione dello stupefacente DALL·E 2 e altri progetti come **Make-A-Scene** (**https://bit.ly/3hBuXfv**) di **Meta** e **Imagen** (**https://imagen.research.google/**) di **Google** stanno esplorando lo spazio della creatività grafica, spingendo verso risultati sempre più spettacolari.

UN PERCORSO DI ANNI

"Alle spalle di DALL-E c'è GPT-3, un modello linguistico autoregressivo che utilizza il deep learning per generare testi"

» QUESTIONI ETICHE E MORALI: COSA SUCCEDE ORA?

La possibilità di generare contenuti automatici apre le porte a una serie di considerazioni etiche e i creatori di **DALL·E 2** si sono in parte già attrezzati. Per esempio, hanno limitato la capacità del sistema di generare immagini violente, che incitano all'odio oppure per adulti. Rimuovendo i contenuti più espliciti dai dati di addestramento, hanno ridotto al minimo l'esposizione di DALL-E 2 a questi concetti. Utilizzano anche tecniche avanzate per evitare generazioni fotorealistiche di volti di individui reali, compresi quelli di personaggi pubblici. Inoltre, la loro **policy** non consente agli utenti di generare i contenuti citati e DALL-E 2 non produce le immagini se i suoi filtri individuano richieste di testo e caricamenti di immagini che potrebbero violare le sue regole. Ci sono infine sistemi di monitoraggio automatizzati e umani per prevenire gli abusi. C'è poi anche il problema del **copyright** stesso: i sistemi di IA vengono allenati su dataset esistenti, quindi DALL·E 2 impara a creare arte... copiando altra arte? La domanda è più che legittima e se la stanno ponendo in tanti nel settore. Per ora non c'è una risposta ma è chiaro che questo sarà un campo fertile per gli avvocati nel prossimo futuro!

Is DALL-E's art borrowed or stolen?

Creative AIs can't be creative without our art.

Daniel Cooper di **Engadget** si è proprio recentemente chiesto se l'arte di **DALL·E 2** è presa in prestito o rubata. Vedi **https://engt.co/3tfNO2i**

Smontare le canzoni pezzo per pezzo

Straordinarie applicazioni dell'Intelligenza Artificiale permettono di separare i brani nei loro componenti musicali

Avete mai pensato "Caspita, che assolo di chitarra favoloso" o "Che bellissima linea vocale" di una delle vostre canzoni preferite? Molto probabilmente lo avete fatto e ora potete anche scoprirli meglio...
Ci sono infatti dei sistemi basati sull'IA che permettono di separare gli **stem** o **steli musicali** in un brano. Per esempio, su **splitter.ai** potete caricare una canzone e farla suddividere automaticamente in due o cinque stem, rispettivamente per parti vocali e strumenti e per voce, basso, percussioni, piano e altri strumenti.

Spleeter di **Deezer** può estrarre voce, batteria, basso e altre tracce da un brano musicale

Un servizio analogo viene fornito da **ezstems.com**. Entrambi i progetti sono basati su **Spleeter**, un sistema Open Source di **separazione delle fonti musicali** basato sull'Intelligenza Artificiale. Potete scoprire come usarlo voi stessi nel **box** qui sotto.

IA per la separazione delle fonti

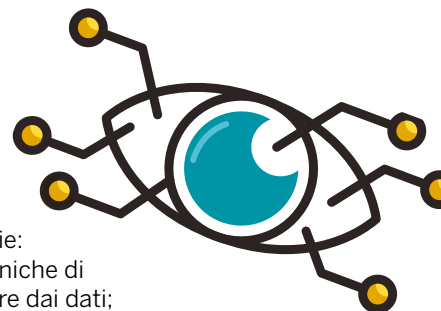
Un altro interessante progetto Open Source su questo tema è **Demucs** (**https://bit.ly/3tpH9Tb**) di **Meta Research**. Semplificando, la separazione delle fonti musicali è l'uso della tecnologia per scomporre un brano nei suoi contributi costitutivi, come appunto la voce, il basso e la batteria. Per il nostro cervello è un'operazione istintiva distinguerli, ma insegnarlo a un computer non è un compito facile. Le tecniche più comunemente utilizzate dall'IA si basano sull'analisi degli **spettrogrammi**, che sono visualizzazioni simili a **mappe di calore** delle diverse frequenze audio di un brano. Questi metodi, tuttavia, hanno dei limiti perché cercano di incasellare i suoni in una matrice predeterminata di frequenza e tempo. Demucs usa invece un modello basato sulle forme d'onda che rileva i loro schemi e poi aggiunge una struttura di livello superiore, in modo simile alla **visione computerizzata** che utilizza le **reti neurali** per rilevare gli schemi di base prima di dedurre quelli più complessi.

» SEPARATE LE CANZONI SUL VOSTRO COMPUTER

Naturalmente non siete obbligati a lavorare sui brani audio usando un servizio terzo, anzi potete fare tutto su un vostro server o sul vostro PC. A questo scopo partite dal **repository** su **GitHub https://github.com/deezer/spleeter**, da cui avete la possibilità di clonare tutto quello che vi serve. **Spleeter** è una libreria per la separazione delle fonti di **Deezer** scritta in **Python** che utilizza **Tensorflow**. Semplifica l'addestramento di un modello di separazione delle fonti (se si dispone di un **dataset** di fonti isolate) e ne fornisce uno all'avanguardia già addestrato per eseguire separazioni di vario tipo: quella di voce (parte cantata) / accompagnamento (**2 stem**), oppure voce / batteria / basso / altro (**4 stem**) e anche voce / batteria / basso / pianoforte / altro (**5 stem**). Spleeter è anche molto rapido, in quanto è in grado di eseguire la separazione dei file audio in **4 stem** a una velocità **100** volte superiore rispetto al tempo reale quando viene eseguito su una **GPU**. Spleeter può essere lanciato dalla **riga di comando**, installato con **pip** o utilizzato con **Docker**. Usarlo è molto semplice:

```
# installare le dipendenze con conda
conda install -c conda-forge ffmpeg
libsndfile
# installare spleeter con pip
pip install spleeter
# separare l'audio in due componenti
spleeter separate -p spleeter:2stems -o
output audio
```

Fate pratica con l'analisi del sentimento

È uno dei progetti di machine learning e IA più diffusi ma è anche la base per molte tecniche popolari

La sentiment analysis, detta anche opinion mining, è uno strumento essenziale per monitorare e comprendere il "sentiment" in tutti i tipi di dati. Analizzando automaticamente i feedback dei clienti, come le opinioni espresse nelle risposte ai sondaggi e nelle conversazioni sui social media, i marchi possono capire cosa li rende felici o insoddisfatti, in modo da poter adattare i prodotti e i servizi alle loro esigenze. Per esempio, si può confrontare il sentiment di un trimestre con quello del successivo per capire se è necessario intervenire. La sentiment analysis si concentra sulla **polarità** di un testo (positiva, negativa, neutra), ma va anche oltre per rilevare sentimenti ed emozioni specifiche (arrabbiato, felice, triste, ecc.), l'urgenza (urgente, non urgente) e persino le intenzioni dell'utente (interessato/non interessato).

Gli algoritmi delle opinioni

La sentiment analysis funziona grazie ad algoritmi di elaborazione del linguaggio naturale (**Natural Language Processing** o **NLP**) e di apprendimento automatico che consentono di determinare automaticamente il tono emotivo delle conversazioni online. Gli **algoritmi** di analisi del sentiment si dividono in tre categorie:
- **Automatici**: i sistemi si basano su tecniche di apprendimento automatico per imparare dai dati;
- **Basati su regole**: questi sistemi eseguono automaticamente la sentiment analysis sulla base di un insieme di regole create manualmente;
- I sistemi **ibridi** combinano approcci basati su regole e automatici.

Come capire il sentimento

Innanzitutto, il sentiment può essere definito utilizzando **5 livelli** che vanno da molto negativo a molto positivo, come le stelle da **1** a **5** di **Google**, **TripAdvisor**, ecc. La sentiment analysis per il rilevamento delle emozioni consente di andare oltre la polarità per rilevare emozioni come felicità, frustrazione, rabbia e tristezza utilizzando lessici o complessi algoritmi di apprendimento automatico. Infine, è utile sapere quali aspetti o caratteristiche particolari vengono citati in modo positivo, neutro o negativo. È qui che l'analisi del sentiment basata sugli **aspetti** può essere d'aiuto. Per esempio, nel caso di "Il piatto è stato servito troppo freddo", vogliamo scoprire cosa pensa il cliente del piatto stesso. Nelle prossime pagine vi cimenterete in questa operazione.

» LA BORSA DEGLI ATTREZZI DEL PROGRAMMATORE IA

Spesso l'Intelligenza Artificiale sembra un soggetto esoterico ma la verità è che per lavorarci servono pochissimi strumenti e risorse tutto sommato limitate. Anzi, in alcuni casi non è nemmeno necessario essere programmatori esperti per ottenere ottimi risultati. Tutto ciò che serve per iniziare a lavorare con un progetto di Intelligenza Artificiale di fatto è **Python**, nella sua versione più recente. Tutto il resto può essere installato successivamente come libreria a seconda del progetto da realizzare. Le due più popolari sono probabilmente **TensorFlow** e **PyTorch**. Se si lavora con l'IA, è necessario eseguire qualche tipo di calcolo di apprendimento profondo. TensorFlow fa proprio questo e potete eseguirlo sia sulla **CPU** sia sulla **GPU**. Si avvale di un sistema di **hub multistrato** che consente di configurare e addestrare rapidamente sistemi neurali con enormi insiemi di dati. È lo strumento, per esempio, che permette a **Google** di riconoscere i testi nelle fotografie. PyTorch è un sistema di Intelligenza Artificiale creato da **Facebook**. Il suo codice è accessibile su **GitHub** e al momento conta più di **60.000** stelle. Offre due funzionalità di alto livello: **calcolo tensoriale** (come **NumPy**) con forte accelerazione tramite GPU e **reti neurali profonde**. Potete anche riutilizzare i vostri pacchetti Python preferiti (come **NumPy**, **SciPy** e **Cython**) per estendere PyTorch quando necessario. Naturalmente non dovete per forza fare tutto da voi, ma di questo parleremo nelle pagine finali di questo articolo.

Imparate a capire i sentimenti online

Mettete le mani nel codice e create un progetto completo per identificare il sentiment nei messaggi su Twitter

Per questo tutorial prenderemo ispirazione dagli articoli degli esperti più influenti in materia e useremo la lingua inglese semplicemente perché è la più semplice per il primo vostro progetto e c'è molto materiale. Preparerete un dataset di tweet campione dal pacchetto NLTK per NLP con diversi metodi di pulizia dei dati. Una volta che il set di dati sarà pronto per l'elaborazione, addestrerete un modello su tweet preclassificati e lo utilizzerete per classificare i tweet campione in sentiment negativi e positivi. Per prima cosa, installate il pacchetto **NLTK** con il gestore di pacchetti **pip**:

```
pip install nltk==3.3
```

Avviate una sessione interattiva di **Python** eseguendo il seguente comando:

```
python3
```

Quindi, importate il modulo **nltk** nell'**interprete python** e scaricate i **tweet** di esempio dal pacchetto NLTK e un modulo che aiuta a **tokenizzare** le parole e le frasi (vedi sotto):

```
import nltk
nltk.download('twitter_samples')
nltk.download('punkt')
```

I tweet negativi e positivi verranno utilizzati per addestrare il modello di analisi del sentiment più avanti nel corso dell'esercitazione. I tweet privi di sentiment saranno invece usati per testare il modello. Se volete utilizzare un vostro **dataset**, potete raccogliere i tweet di un periodo di tempo specifico, di un utente o di un **hashtag** utilizzando l'**API** di **Twitter**. Dopo aver importato NLTK e scaricato i tweet di esempio, uscite dalla sessione interattiva digitando `exit()`. Siete ora pronti a importare i tweet e a iniziare l'elaborazione dei dati. Il linguaggio nella sua forma originale non può però essere processato con precisione da una macchina, quindi è necessario elaborarlo per renderlo più comprensibile al computer. Il primo passo per dare un senso ai dati è un processo chiamato tokenizzazione, ovvero la suddivisione delle stringhe in parti più piccole chiamate **token**. Per iniziare, create un nuovo file **nlp_lxp.py** per contenere lo **script**. In questo file, importerete innanzitutto **twitter_samples**, in modo da poter lavorare con quei dati:

```
from nltk.corpus import twitter_samples
```

Quindi, create variabili per tweet_positivi, tweet_negativi e tweet_neutrali:

```
from nltk.corpus import twitter_samples
tweet_positivi = twitter_samples.strings('tweet_positivi.json')
tweet_negativi = twitter_samples.strings('tweet_negativi.json')
tweet_neutrali = twitter_samples.strings('tweets.20150430-223406.json')
```

NLTK fornisce un **tokenizzatore** predefinito per i tweet con il metodo `.tokenized()`. Aggiungete una riga per creare un oggetto che tokenizzi il dataset **tweet_positivi.json**:

```
tweet_tokens = twitter_samples.tokenized('tweet_positivi.json'
```

Per testare la tokenizzazione potete procedere con:

```
print(tweet_tokens[0] [0])
```

che dà un risultato di questo tipo:

```
['#opensource',
 '@Linux',
 'Open',
 'Source',
 'è',
 'il',
 'top',
 ':)']
```

Qui il metodo `.tokenized()` restituisce caratteri speciali come **@** e **_**. In seguito, saranno rimossi attraverso **espressioni regolari**.

Normalizzazione dei dati

La normalizzazione aiuta a raggruppare parole con lo stesso significato ma con forme diverse. Senza di essa, "mangiare", "mangia" e "mangiando" verrebbero trattate come parole diverse, anche se probabilmente vi sarebbe più utile che venissero trattate come un unico termine. Qui si utilizzerà il processo di **lemmatizzazione**, che normalizza le parole con il contesto del vocabolario e l'**analisi morfologica** dei vocaboli nel testo. L'algoritmo di lemmatizzazione analizza la struttura della parola e il suo contesto per convertirla in una forma normalizzata. Scaricate i dati necessari:

NEGATIVO!

La maggior sfida della sentiment analysis è la complessità del linguaggio. La negazione ha un'influenza primaria sulla polarità contestuale delle parole e dei testi d'opinione. Frasi di negazione come mai, nessuno, niente e altre possono invertire le polarità delle "parole d'opinione". Questo a volte è difficile da rilevare e può ridurre la precisione del motore.

```
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
```

Prima di eseguire un **lemmatizzatore**, è necessario determinare il contesto di ogni parola del testo. Questo si ottiene con un algoritmo di **tagging**, che valuta la posizione relativa di una parola in una frase. Per farlo, si può partire da una sessione Python:

```
from nltk.tag import pos_tag
    from nltk.corpus import twitter_samples
    tweet_tokens = twitter_samples.tokenized('tweet_
positivi.json')
    print(pos_tag(tweet_tokens[0]))
```

Vedrete ogni parola contrassegnata da un **tag**. In generale, se un tag inizia con **NN**, la parola è un nome e se inizia con **VB** è invece un verbo. L'elenco completo dei tag è disponibile sul sito **https://bit.ly/3TtLDmg**. Per incorporare questa operazione in una funzione che normalizza una frase bisogna prima generare i tag per ogni token del testo e poi lemmatizzare ogni parola usandoli. Aggiornate il file Python con la seguente funzione che lemmatizza le frasi:

```
def lemmatize_sentence(tokens):
    lemmatizer = WordNetLemmatizer()
    lemmatized_sentence = []
    for word, tag in pos_tag(tokens):
        if tag.startswith('NN'):
            pos = 'n'
        elif tag.startswith('VB'):
            pos = 'v'
        else:
            pos = 'a'
        lemmatized_sentence.append(lemmatizer.
lemmatize(word, pos))
    return lemmatized_sentence
```

Se lo chiudete e lo lanciate, noterete che i verbi vengono sostituiti con la loro **forma radicale** e i sostantivi plurali con quelli singolari. L'ultima parte della normalizzazione è la **rimozione del rumore**. Per questo non forniremo alcun codice, perché dipende dal contesto. Per esempio, potreste voler rimuovere i caratteri speciali, la punteggiatura, ecc. Un elemento che probabilmente vorrete rimuovere, indipendentemente dal contesto, sono le **stop word**. Sono generalmente irrilevanti nell'elaborazione del linguaggio, a meno di casi d'uso specifici che ne giustifichino l'inclusione. Stiamo parlando di parole come "è", "il", ecc. In una sessione Python, eseguite:

```
nltk.download('stopwords')
```

E poi nel codice potete scrivere:

```
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
```

Per creare un **array** di `stop_words` da usare per filtrare i token. Infine, create una funzione `remove_noise` che se ne occupi.

Preparazione all'analisi

Innanzitutto, si preparano i dati da inserire nel modello. Per svolgere l'esercizio di modellazione si utilizzerà il classificatore **Naive Bayes** di NLTK. Si noti che il modello non richiede solo un elenco di parole in un tweet, ma un dizionario Python con le parole come chiavi e **True** come valori. La funzione che segue crea una funzione generatore per cambiare il formato dei dati puliti. I dizionari corrispondenti sono memorizzati in **positive_tokens_for_model** e **negative_tokens_for_model**. Aggiungete questo codice per creare i dizionari e preparare i dati per l'addestramento della classe `NaiveBayesClassifier`.

```
import random
def get_tweets_for_model(cleaned_tokens_list):
    for tweet_tokens in cleaned_tokens_list:
        yield dict([token, True] for token in tweet_tokens)
positive_tokens_for_model = get_tweets_for_
model(positive_cleaned_tokens_list)
negative_tokens_for_model = get_tweets_for_
model(negative_cleaned_tokens_list)
positive_dataset = [(tweet_dict, "Positivo")
            for tweet_dict in positive_tokens_for_model]
negative_dataset = [(tweet_dict, "Negativo")
            for tweet_dict in negative_tokens_for_model]
dataset = positive_dataset + negative_dataset
random.shuffle(dataset)
train_data = dataset[:7000]
test_data = dataset[7000:]
```

Questo codice attribuisce un'etichetta positiva o negativa a ogni tweet per poi creare un set di dati unendo i tweet positivi e negativi. Per evitare parzialità, è stato aggiunto del codice per disporre i dati in modo casuale, utilizzando il metodo `.shuffle()` di `random`. Infine, il codice divide i dati mescolati in un rapporto di **70 a 30** per l'addestramento e il test, rispettivamente. Questo è utile per avere un buon equilibrio fra materiale di verifica e contenuti da analizzare.

Testare il modello

Infine, potete usare la classe `NaiveBayesClassifier` per costruire il modello. Utilizzate il metodo `.train()` per addestrarlo e il metodo `.accuracy()` per testarlo sui dati di prova.

```
from nltk import classify
from nltk import NaiveBayesClassifier
classifier = NaiveBayesClassifier.train(train_data)
```

Verificate poi come si comporta il modello su tweet scelti a caso da Twitter. Aggiungete il codice che segue al file:

```
ffrom nltk.tokenize import word_tokenize
custom_tweet = "Sono stato in questo ristorante solo
una volta: i piatti erano freddi e il servizio pessimo"
custom_tokens = remove_noise(word_tokenize(custom_
tweet))
print(classifier.classify(dict([token, True] for token in
custom_tokens)))
```

Eseguite lo script per analizzare il testo personalizzato. Ecco l'output per quello dell'esempio qui sopra:

```
Output
'Negative'
```

Ecco fatto! Avete costruito correttamente un modello in grado di identificare le emozioni nei tweet. Benvenuti nel mondo dell'IA!

PAROLINE UTILI

Le stop word vanno in genere ignorate, come abbiamo visto nel tutorial, ma non sempre è così. In base al contesto e alla lingua, alcune parole brevi possono risultare rilevanti. Per esempio, il vostro progetto potrebbe avere bisogno di distinguere tra singolare e plurale, quindi la differenza tra "quello" e "quelli" diventa importante.

CHE IRONIA

Gli esseri umani sono complessi e a volte diciamo una cosa ma intendiamo il contrario. "Siamo arrivati a cena a mezzogiorno e siamo stati serviti alle 13: un servizio molto veloce, vero?" non è un complimento. Abituatevi a modelli che a volte sbagliano con questo tipo di ironia e sarcasmo.

Le librerie essenziali

A prescindere dal progetto che intendete realizzare, alcuni componenti di Python sono utilissimi: ecco quali vi serviranno praticamente sempre

Python è il linguaggio più utilizzato per l'apprendimento automatico e per i progetti di Intelligenza Artificiale in generale. Una sua caratteristica fondamentale che attira molti utenti è la sua vasta collezione di librerie Open Source. Possono essere utilizzate da programmatori di ogni livello di esperienza per attività di ML e IA, scienza dei dati, manipolazione di immagini e dati e molto altro ancora. Ecco una panoramica di quelle che userete di più.

PyTorch

PyTorch è un **framework** per l'apprendimento automatico basato su **Torch** (**torch.ch**). Questa libreria per la scienza dei dati può creare grafici computazionali modificabili durante l'esecuzione del programma ed è spesso utilizzata per applicazioni di Machine Learning e Deep Learning. Veloce e flessibile, dispone di potenti **API** e di un **toolkit** per il **linguaggio naturale**.

TensorFlow

TensorFlow (**https://www.tensorflow.org/**) può essere utilizzato per una serie di attività nell'ambito dell'apprendimento automatico e dell'Intelligenza Artificiale ma si concentra in particolare sull'addestramento e l'**inferenza** delle reti neurali profonde. È stato sviluppato dal team di **Google Brain** per uso interno di **Google** nella ricerca e nella produzione e reso disponibile sotto la **licenza Apache 2.0** nel **2015**. Offre un'architettura flessibile, funziona su una varietà di piattaforme computazionali e ha **capacità di astrazione**. Questa caratteristica consente agli sviluppatori di concentrarsi sulla logica complessiva dell'applicazione invece di occuparsi dei dettagli dell'implementazione degli algoritmi.

TensorFlow di **Google** è un fido compagno per ogni tipo di progetto

NumPy

NumPy (**https://numpy.org/**) è una libreria numerica molto utilizzata che offre funzioni matematiche complete, generatori di numeri casuali, **routine** di algebra lineare, trasformate di **Fourier** e altro ancora. Il suo nucleo è in **C**, il che consente di godere della flessibilità di Python con la velocità del **codice compilato**.

Pandas

Pandas (**https://pandas.pydata.org**/) offre un'ampia gamma di strumenti per la manipolazione e l'analisi dei dati. Permette di leggere dati da numerose fonti (come **CSV**, **database SQL** e **file JSON**) e di lavorare facilmente con dati multidimensionali strutturati e **serie temporali**. Include strumenti per l'**indicizzazione** e l'allineamento dei dati e per la fusione di insiemi di dati.

Keras

Keras (**https://keras.io/**) semplifica la progettazione e lo sviluppo di una rete neurale per chi si avvicina all'apprendimento automatico, anche se può essere usato in programmi molto avanzati ed è utilizzato da organizzazioni scientifiche come il **CERN** e la **NASA**. La libreria offre dei componenti fondamentali con cui potete sviluppare modelli complessi di apprendimento automatico e la sua interfaccia, concepita dichiaratamente prima per gli uomini e poi per le macchine, facilita e velocizza il lavoro.

Scikit-learn

Scikit-learn (**scikit-learn.org/stable**) offre una serie di strumenti per l'**analisi predittiva** dei dati. Comprende un'ampia gamma di algoritmi di **clustering**, **regressione** e classificazione che possono essere utilizzati in molte applicazioni come il rilevamento di **spam**, il riconoscimento di immagini, la creazione di previsioni e la segmentazione dei clienti.

LIBRERIE OPEN SOURCE
"Possono essere utilizzate da programmatori di ogni livello per attività di ML, IA e scienza dei dati"

Servizi cloud di IA pronti all'uso

Non sempre potete fare tutto da voi o in alcuni casi è più economico usare API esterne: ecco quali valgono davvero la pena

Creare strumenti nell'ambito dell'IA non è facilissimo o veloce: in alcuni casi servono letteralmente centinaia di milioni di euro di investimenti. Per questo motivo spesso è utile, più che scrivere del codice e allenare un modello, trovare i servizi giusti che al prezzo più competitivo fanno quello che vi serve nelle vostre applicazioni sfruttando l'Intelligenza Artificiale. In questa pagina vediamo alcuni dei più utili e soprattutto quelli che difficilmente potreste creare voi da zero. Non sono Open Source ma con il loro aiuto potete sicuramente creare progetti che lo sono.

Un doppiaggio professionale

La tecnologia della sintesi vocale, che consente di convertire il testo in parlato, esiste da tempo, ma è spesso associata a risultati "robotici" e poco naturali. Con **Murf Studio** (**murf.ai**) si possono invece realizzare voiceover realistici, con oltre **120 voci basate sull'IA** in più di **20 lingue**. È inoltre possibile regolare e sincronizzare i file audio con i video o le immagini all'interno del servizio.

Spolpare il testo

Amazon Comprehend è un servizio di **elaborazione del linguaggio naturale** (**NLP**) che utilizza il machine learning per svelare informazioni dettagliate e collegamenti preziosi all'interno dei testi. Per esempio, automatizza l'estrazione di informazioni dettagliate da pacchetti di documenti legali come contratti e atti giudiziari. Inoltre classifica ed estrae entità da documenti di servizi finanziari, come richieste di risarcimento assicurativo o pacchetti di mutui, oppure trova relazioni tra eventi finanziari in un articolo sul tema.

Trovare errori nel codice

Programmare nasconde naturalmente tanti trabocchetti e l'IA sta facendo passi da gigante per venirvi incontro. Sono ormai numerosi i servizi che si occupano di analizzare il vostro codice e trovare errori e potenziali problemi di sicurezza, uno fra tutti **www.deepcode.ai**.

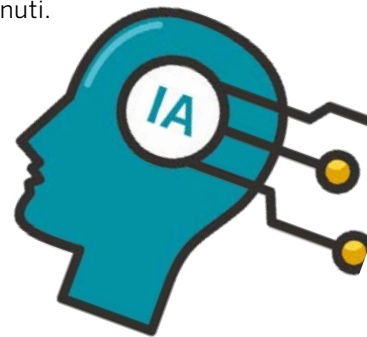
Strumenti per qualsiasi progetto

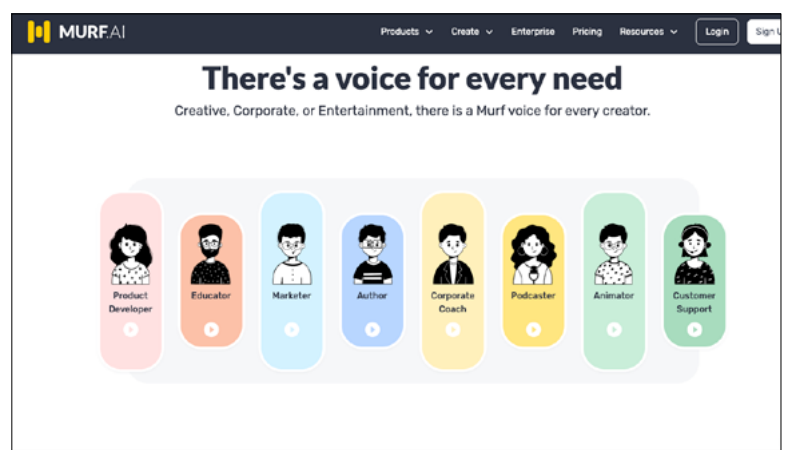
Google è all'avanguardia nell'IA (come abbiamo visto il mese scorso) e, se vi sentite a vostro agio nell'usare i suoi servizi, c'è una serie di sistemi per accelerare la creazione di modelli all'indirizzo **https://cloud.google.com/products/ai**.

Trascrizioni perfette

Transcribe Call Analytics di **AWS** (**Amazon Web Services**) consente di estrarre in modo rapido ed efficiente informazioni dalle conversazioni con i clienti o da altri brani parlati, in modo da poter poi eseguire vari tipi di analisi sui loro contenuti. Chi produce o distribuisce contenuti multimediali può per esempio utilizzare **Amazon Transcribe** per convertire automaticamente le risorse audio e video in archivi completamente ricercabili. Si possono anche creare servizi di sottotitolazione **on-demand**.

Un OCR molto intelligente

Sono diversi anche i servizi di riconoscimento dei caratteri con un'anima basata sull'Intelligenza Artificiale. Uno di questi è **https://rossum.ai/dp/ocr-machine-learning** che risulta perfetto per compiti molto difficili e delicati come per esempio riconoscere correttamente il contenuto delle fatture. LXP

Con **Murf Studio** si possono usare voci di sintesi realistiche per creare **podcast** e doppiaggi di qualità per video e presentazioni

Crediti icone articolo: Freepik

PLASMA BIGSCREEN

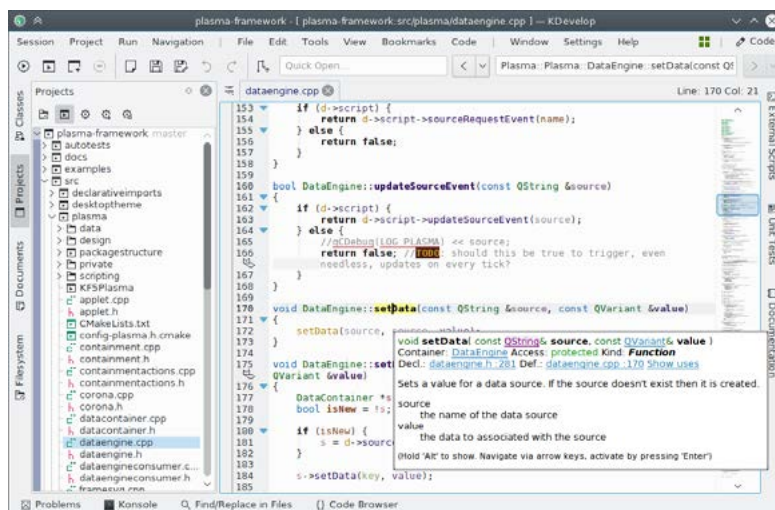
Scoprite come avere tutta la libertà, la privacy e la personalizzabilità di Linux sul vostro televisore e gli strumenti per creare nuove applicazioni

Plasma è l'ambiente desktop di KDE ed è progettato per essere facile da usare, pur offrendo una notevole potenza. Secondo le parole dei suoi sviluppatori è "l'ambiente desktop più personalizzabile che esista" e l'obiettivo guida dichiarato della comunità KDE è quello di renderlo semplice nelle sue scelte predefinite e completo quando necessario. Da poco uscito nell'edizione 5.26, è presente in molte distribuzioni per computer desktop e dispositivi mobili ed è in corso di sviluppo la sua versione per schermi grandi, chiamata Plasma Bigscreen. Progettato per essere utilizzato su televisori, monitor e altri schermi di grandi dimensioni, **Plasma Bigscreen** è dotato di un'interfaccia utente di **10 piedi** (equivalenti a **120 pollici**) che è stata studiata per garantire che possiate vedere in modo ottimale tutto ciò che viene visualizzato sullo schermo da lontano, come per esempio dal divano di un classico salotto. Inoltre l'ambiente è controllabile tramite comandi vocali o il telecomando del televisore e c'è anche una serie di applicazioni studiate per essere eseguite su di esso. Lo scopo di questo progetto Open Source (**https://plasma-bigscreen.org/it/**) è offrire un ecosistema rispettoso della privacy e sicuro per televisori o **set-top box**. L'obiettivo finale è trasformarli in dispositivi completamente liberi. Le linee guida per l'interfaccia umana (**HIG**) di **KDE** (**https://develop.kde.org/it/hig/**) adottati da Plasma lo rendono ideale per l'utilizzo su **televisori smart** o collegati a **Raspberry Pi** per renderli tali. Incarna infatti i due principi chiave di KDE: semplice per scelta, potente quando serve. La facilità d'uso, essenziale per un dispositivo controllato con la voce o il telecomando, si basa sull'idea di ridurre al minimo gli elementi a schermo non cruciali per l'attività principale e di rivelare informazioni o funzioni aggiuntive solo quando necessario. La potenza, che include la libertà di personalizzazione che rende i sistemi **Linux** unici, è centrata invece sul principio di fornire impostazioni predefinite ragionevoli ma considerare funzionalità opzionali per chi può averne bisogno.

L'unione fa la forza

Plasma Bigscreen utilizza una serie di strumenti software Open Source per offrire funzionalità di avanguardia e un sistema operativo versatile. Sfrutta, per esempio, **MyCroft AI** per consentire agli utenti di controllare vocalmente il televisore, **libCEC** per renderlo compatibile con i telecomandi e **Pulseaudio** per gestire il sistema audio. Il primo componente su cui si basa Plasma è la piattaforma di KDE (**Frameworks**, **https://develop.kde.org/it/**) per lo sviluppo, basata sulla libreria per la creazione di applicazioni **Qt**. I **framework** di KDE sono librerie aggiuntive per la creazione di applicazioni con Qt. Ciascuno di essi è stato testato in scenari reali e tutti offrono un'ampia documentazione. Le loro **API** hanno inoltre uno stile comune che risulterà familiare a chi ha già sviluppato con Qt. I framework sono prodotti seguendo il collaudato modello operativo di KDE con un calendario di rilascio prevedibile, un processo di contribuzione chiaro e che non favorisce nessun produttore, una governance aperta e licenze flessibili **LGPL** o **MIT**. I framework sono multipiattaforma e funzionano su **Windows**, **Mac**, **Android** e **Linux**. Uno dei principali strumenti che KDE mette a disposizione dei programmatori è **KDevelop**, un **IDE** multipiattaforma per **C**, **C++**, **Python**, **QML/JavaScript** e **PHP** con licenza **GNU GPL**. Il cuore di KDevelop è la combinazione di un **editor** avanzato con l'**analisi semantica** del codice, che aiuta nella programmazione. KDevelop offre diversi flussi di lavoro per assistere durante il processo di sviluppo. Contribuisce a migliorare la qualità del codice, a verificarne la funzionalità e a implementarlo sulle diverse piattaforme.

Kirigami per la convergenza

Un framework particolarmente importante per Plasma Bigscreen è **Kirigami UI** (**https://develop.kde.org/frameworks/kirigami/**), pensato per facilitare la produzione di **applicazioni convergenti** "che funzionano su telefoni, TV e su tutto quello che c'è in mezzo". L'espressione "applicazione convergente", in questo contesto, indica che le persone possono utilizzare facilmente il programma indipendentemente dal dispositivo su cui viene eseguito. Se, per esempio, viene usata su un computer desktop, l'applicazione si adatta allo schermo grande e a mouse e tastiera; se è sfruttata su un cellulare, accetta **input** da **touchscreen** e si adegua a uno schermo verticale più piccolo. Le applicazioni Kirigami si adattano non solo ai dispositivi su cui vengono installate, ma anche al modo in cui vengono utilizzate, cercando sempre di offrire un **layout** ottimale. Kirigami è costruito sulla base del **linguaggio QML** e dei componenti **Qt Quick Controls 2** del progetto **Qt**. QML supporta i sistemi **touch**, il che lo rende ideale per le applicazioni mobili. I componenti Qt Quick sono elementi visivi riutilizzabili che si possono usare per costruire le interfacce delle applicazioni e Kirigami ne raccoglie numerosi

KDevelop è un ambiente di sviluppo integrato (**IDE**) multipiattaforma che fa parte degli strumenti per programmatori di **KDE**

» LE APPLICAZIONI DI PLASMA BIGSCREEN

Plasma 5.26, uscito l'11 ottobre, ha portato due nuove applicazioni per **Bigscreen**: **Aura** e **Plank Player**. Il primo è un browser progettato per facilitare la navigazione in Internet dal divano di casa con il telecomando, che potete usare come un **puntatore laser** per "fare click" sui **link**. Plank Player è invece un lettore multimediale, semplice nelle funzioni offerte e nell'uso, che consente di riprodurre video da un dispositivo collegato al televisore, controllando con un telecomando. Se invece volete cercare video sul Web, avete a disposizione **PeerTube for Plasma Bigscreen** o **YouTube Bigscreen** che si possono gestire con comandi vocali. Se siete a caccia di musica, potete optare per **SoundCloud Player for Plasma Bigscreen**, che vi dà accesso vocale alla nota piattaforma di **streaming** di brani musicali e **podcast**. Volete sapere qual è la data di nascita di **Linus Torvalds**? Potete chiedere e ottenere informazioni su una persona da **Wikidata** con **WikiData Support for Plasma Bigscreen**. In genere le applicazioni funzionano abbastanza bene, ma se avete qualche problema con il controllo vocale oppure con il telecomando potete trovare delle risposte utili all'indirizzo: **https://plasma-bigscreen.org/faq/.**

Plank Player è un lettore multimediale per la visualizzazione di file locali su **Plasma Bigscreen** utilizzabile con un semplice telecomando

progettati per creare applicazioni convergenti. Mentre per produrre la parte visiva della applicazione viene usato QML, la logica del codice che si occupa delle sue funzionalità viene solitamente creata in **C++** per motivi di prestazioni. Per creare un'applicazione Kirigami sono necessari un **compilatore C++**, i pacchetti di sviluppo Qt e il framework stesso. Potete trovare un **tutorial** su come fare i primi passi nello sviluppo in Kirigami all'indirizzo **https://bit.ly/3MErhoO**. Se conoscete un po' di **JavaScript**, allora gran parte di QML vi sembrerà familiare, anche se ha le sue peculiarità. La documentazione di Qt (**https://doc.qt.io/qt-5/qtqml-index.html**) contiene un'ampia quantità di materiale su questo linguaggio, se avete voglia di cimentarvi nello sviluppo.

Quattro chiacchiere con Mycroft

Per consentire di controllare il televisore con la voce, Plasma Bigscreen usa **Mycroft AI** (**https://github.com/MycroftAI/**). È il principale assistente vocale Open Source sul mercato. È simile ad **Amazon Alexa**, **Google Assistant** o **Siri di Apple**, ma offre molto più controllo e flessibilità, essendo completamente customizzabile grazie al codice Open Source. I suoi sviluppatori ritengono che un assistente vocale debba essere flessibile, personalizzabile, non vincolato a un produttore e incentrato sulla privacy, caratteristiche che quelli proprietari presenti oggi sul mercato secondo loro non offrono. La piattaforma aperta di Mycroft si differenzia da quelle proprietarie per la privacy, la personalizzazione e il fatto che i suoi dati sono aperti. Innanzitutto elimina le **query** in tempo reale e i dati degli utenti non vengono estratti, aggregati, elaborati o venduti. Offre inoltre molte opzioni di personalizzazione, come la modifica della **wake word** (la parola che attiva il riconoscimento vocale, come **Alexa** nei sistemi **Amazon**), della voce usata dal dispositivo e persino dell'esperienza utente. Chi programma può sviluppare le proprie **skill** (le applicazioni vocali che forniscono diverse funzionalità agli utenti) con maggiore libertà rispetto ad altre piattaforme. Infine, Mycroft pubblica i dati degli utenti che hanno deciso di contribuire al progetto permettendo di registrare la loro voce. Questo **dataset** aperto viene utilizzato per migliorare l'individuazione delle **wake word**, la trascrizione del parlato, la comprensione del linguaggio naturale e la sintesi vocale. Il software funziona su computer desktop, sull'hardware sviluppato da Mycroft e sulla **Raspberry Pi**. Attualmente sono disponibili diverse applicazioni ottimizzate per la TV che vengono fornite con le varie distribuzioni che includono Plasma Bigscreen. Offrono un'ampia gamma di funzioni di base con il controllo vocale, ma il bello di trovarsi in un ecosistema Open Source è che potete sempre sviluppare le vostre.

Il **browser Aura** è stato progettato per facilitare la navigazione in Internet con un telecomando usato come un **puntatore laser**

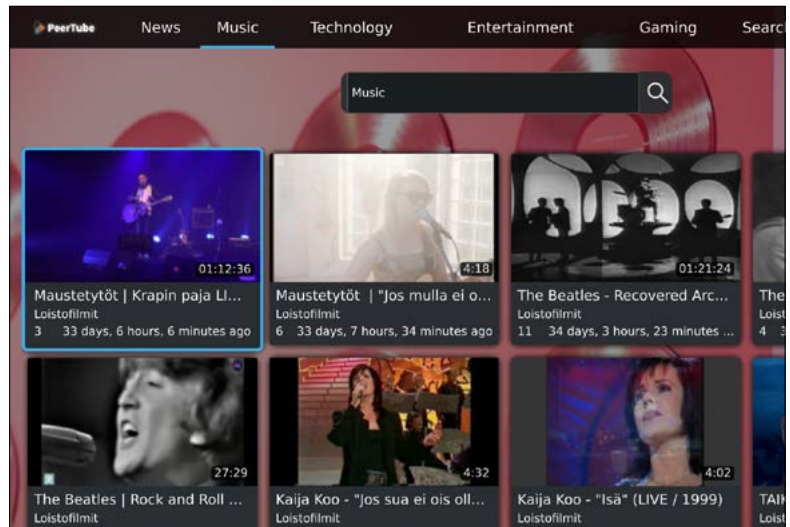
Sviluppare applicazioni vocali

Tutte le applicazioni vocali per Mycroft AI utilizzate su Plasma Bigscreen iniziano come semplici skill vocali con un'interfaccia grafica. Le skill per Mycroft sono scritte utilizzando **Python** e per crearne una dovete avere almeno un'esperienza di base in questo linguaggio di programmazione e un'installazione o un dispositivo Mycroft. Potete eseguire il software su molte distribuzioni, tra cui **Ubuntu/Debian**, **Arch** e **Fedora**. **Clonate** il **repository** all'indirizzo **https://github.com/MycroftAI/mycroft-core** ed eseguite lo **script** di installazione incluso. Esiste anche una versione di Mycroft appositamente progettata per funzionare su **Raspberry Pi 3** o **4**. Si chiama **Picroft** ed è basata sulla versione ufficiale di **Raspbian Buster Lite**. È disponibile come immagine disco pronta per essere masterizzata su una **scheda microSD**. Trovate le istruzioni complete su **https://bit.ly/3CCNkaK**. All'indirizzo **https://bit.ly/3S9CIeB** sono invece riportate le indicazioni di base per creare un'applicazione vocale. Mostra anche, per una semplice skill "Ciao Mondo", un esempio del file **init.py** in cui la maggior parte di essa viene definita:

» UN TELECOMANDO PER LINUX

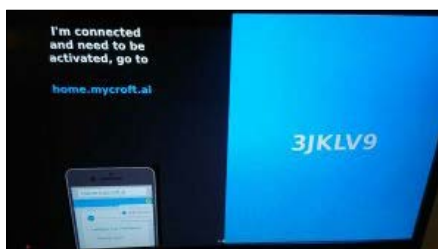
Il **Consumer Electronics Control** (**CEC**) è una funzione del **protocollo HDMI** progettata per controllare diversi dispositivi a esso collegati tramite un unico telecomando ed è ciò che utilizza **Plasma Bigscreen** tramite **libCEC** di **Pulse-Eight**. Quest'ultimo è una **piattaforma di abilitazione** per il **bus CEC** che consente agli sviluppatori di interagire con altri dispositivi HDMI senza doversi preoccupare dell'**overhead** di comunicazione, dell'**handshaking** e dei vari modi di inviare un messaggio adottati dalle diverse aziende. Molti produttori di televisori richiedono che la modalità **HDMI-CEC** sia attivata manualmente e per farlo è necessario consultare il manuale d'uso del dispositivo. L'opzione HDMI-CEC può essere indicata con nomi diversi a seconda dei produttori di TV. Per esempio, **TCL** la chiama **T-Link**, **Panasonic Viera Link**, **Samsung Anynet+** e **Sony Bravia Sync**. Assicuratevi anche che il vostro dispositivo Plasma Bigscreen sia effettivamente supportato da libCEC controllando l'hardware compatibile all'indirizzo **https://github.com/Pulse-Eight/libcec#supported-hardware**. Per maggiori dettagli consultate anche **http://libcec.pulse-eight.com/**.

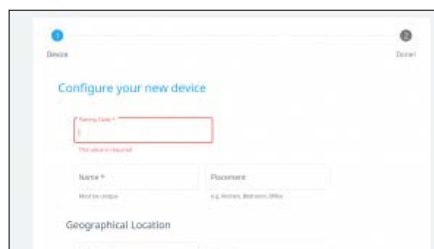
```
import sys
from adapt.intent import IntentBuilder
from mycroft import MycroftSkill, intent_handler

class MyExampleSkill(MycroftSkill):

def __init__(self):
   super().__init__("MyExampleSkill")

def initialize(self):
   self.add_event('myexampleskill.authorname.
home', self.homepage)
   self.gui.register_handler("ExampleSkill.
ButtonClicked", self.handleButtonClick)

def homepage(self):
   self.gui['exampleText'] = "Ciao mondo"
   self.gui.show_page("example.qml")
   self.speak("Benvenuto a Ciao mondo")

@intent_handler('handle_example_helloWorld').
require('example-one')
def handle_example_helloWorld(self, message):
   self.gui['exampleText'] = "Ciao mondo"
   self.gui.show_page("example.qml")
   self.speak("Benvenuto a Ciao mondo")

def handleButtonClick(self):
   self.speak_dialog("example-speak")

def stop(self):
   pass

def create_skill():
   return MyExampleSkill()
```

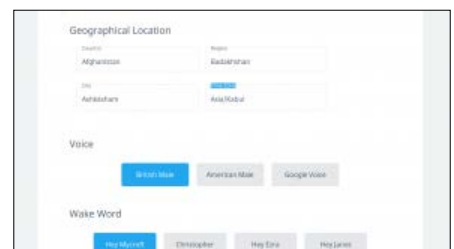
È quindi necessario aggiungere alla propria skill un'interfaccia utente grafica scritta utilizzando il **linguaggio di markup QML**. Potete trovare un aiuto per questa fase all'indirizzo **https://bit.ly/3CGmQ85**. Le skill vocali di Mycroft con interfaccia grafica si possono trasformare facilmente in applicazioni vocali complete, aggiungendo alcuni elementi come una pagina principale e un'icona (vedi **https://bit.ly/3S9CIeB**). Nel complesso, Plasma Bigscreen è destinato a dare agli utenti una libertà, una flessibilità e una privacy senza precedenti. Sebbene sia ancora in fase di sviluppo esistono già distribuzioni che lo includono. L'interfaccia utilizza **KWin** e **Wayland** ed è ora per lo più stabile; è inoltre disponibile un sottoinsieme delle normali funzionalità di **KDE Plasma**. Potete quindi usare **Plasma Bigscreen** e sviluppare per esso sul vostro computer. Perché non provare? LXP

Con l'applicazione vocale di **Peertube** (**https://bit.ly/3TaTJkk**) potete parlare con **Plasma Bigscreen** e chiedergli per esempio di mostrarvi i video di tendenza o quelli del vostro gruppo preferito

DARE UN CONTRIBUTO

"Attualmente sono disponibili diverse applicazioni nelle distribuzioni che includono Plasma Bigscreen ma potete sempre sviluppare le vostre"

ACCOPPIAMENTO DEL SERVER MYCROFT

1 Primo accesso

Quando collegate per la prima volta il vostro dispositivo **Plasma Bigscreen** a Internet, appare questa schermata e, in un paio di minuti, un codice. Per utilizzare il controllo vocale **Mycroft** è necessario accoppiare il dispositivo su **home.mycroft.ai**.

2 Inserire il codice

È possibile farlo da qualsiasi PC o **smartphone**. Dopo aver creato un **account** e aver effettuato l'accesso, visitate il sito **https://account.mycroft.ai/devices** per registrare il vostro dispositivo. Inserite il codice di accoppiamento quando richiesto.

3 Selezionare le opzioni

Assegnate al dispositivo un nome e una posizione. Scegliete la voce e la **wake word** desiderate, poi fate click su **NEXT** per accoppiare il dispositivo. Attendete qualche secondo e vedrete apparire una schermata di conferma.

Crediti: https://plasma-bigscreen.org/manual/

Recensioni

Tutte le novità in campo software e hardware testate e valutate ogni mese dai nostri laboratori

Se vuoi segnalarci qualche novità scrivi a redazione@linuxpro.it

Una breve legenda

Ogni test di questa sezione è accompagnato da un giudizio che riassume con quattro indici numerici le principali qualità dell'applicazione o del prodotto hardware messo alla prova. I laboratori di Linux Pro assegnano un voto da 1 a 10 alle seguenti categorie:

Caratteristiche: fornisce tutte le funzioni di cui abbiamo bisogno? È innovativo?
Prestazioni: esegue in maniera efficiente le sue funzioni? È veloce e affidabile?
Facilità d'uso: dispone di un'interfaccia grafica chiara e facilmente fruibile? La documentazione che lo accompagna è sufficientemente completa ed esaustiva?
Qualità/prezzo: ha un prezzo competitivo? Vale i soldi richiesti per il suo acquisto?

Il nostro giudizio viene poi riassunto da un voto finale, espresso anche graficamente. Ecco la legenda dei voti:

10 Nulla da eccepire. Un prodotto praticamente perfetto.
8-9 Un buon prodotto. I pochi difetti presenti non sono gravi.
6-7 Compie il suo lavoro ma necessita di ulteriori sviluppi.
4-5 Deve migliorare prima di raggiungere un voto sufficiente.
1-3 Un completo disastro. Gli sviluppatori devono tornare alla fase di progettazione.

Ricordiamo infine che i software citati nelle sezioni Confronto e Da non perdere sono spesso presenti nel DVD sotto la voce "Rivista" sotto forma di codice sorgente o binario.

QUESTO MESE...

Space Tail

OpenMV Cam H7 R2

Puppy Linux 9.5

Space Tail

Un platform di avventura in 2.5D che vi fa vivere l'esplorazione dello spazio nei panni e con i sensi di un cane

SPECIFICHE

Minime
OS: Una distro a 64 bit
GPU: GTX 950 / AMD R7 370 o altra con 2 GB di VRAM.
CPU: i5 4460 / AMD FX 4100
Memoria: 6 GB di RAM

Consigliato
GPU: GTX 970 / AMD Radeon RX 570 o altra con 4 GB di VRAM.
CPU: i5 6400 / AMD Ryzen 1700
Memoria: 8 GB di RAM

Ai componenti da **platform game** si aggiungono quelli di avventura, visti dalla inusuale prospettiva di un cane

Sono passati **65** anni da quando la cagnolina **Laika** (che in realtà si chiamava **Kudryavka**) è stata inviata nello spazio per una missione senza ritorno, ottenendo il triste primato di essere il primo animale a orbitare intorno alla Terra. A commemorare il suo sacrificio (e ad aprire la possibilità di un finale migliore alla storia) arriva **Space Tail: Every Journey Leads Home**. Si tratta di un gioco di piattaforme **2.5 D** ("a due dimensioni e mezzo"), ossia che usa varie tecniche grafiche bidimensionali per creare l'illusione della tridimensionalità, ma in cui l'elemento avventura è importante quanto l'azione.

Interagite con i personaggi non giocanti nella modalità relazione, selezionando le azioni da un menu

Annusando lo spazio

Il vostro personaggio è **Bea**, un cane astronauta in missione nelle profondità dello spazio, dove viaggia alla scoperta di altri pianeti, ognuno diverso e con i suoi specifici abitanti, che possono essere amichevoli o meno nei suoi confronti. Il modo in cui interagisce con loro (e comprende il mondo circostante) si basa sul fatto che è un cane e usa i suoi sensi come tale. La sua vista le permette di individuare oggetti interattivi e pericolosi, mentre il suo udito molto sviluppato le consente di rilevare il movimento di nemici e meccanismi al di là della portata degli altri sensi. Anche l'olfatto ha un ruolo importante: viene utilizzato per seguire i nemici, individuare sostanze pericolose e determinare la posizione di oggetti nascosti. Tenendo premuto il pulsante di relazione si avvia un menu contestuale che consente di selezionare uno dei segnali o comportamenti che si desidera attivare quando si interagisce con i personaggi non giocanti. Il titolo è coinvolgente, con una bella grafica e sfide logiche interessanti. Giocare nei panni di un cane offre una prospettiva nuova e divertente, che fa sentire ancora di più il desiderio di proteggere il proprio personaggio. A volte c'è qualche rallentamento nell'azione di gioco e forse ci sono un po' troppi testi, ma nel complesso è un titolo indipendente divertente e originale che vale la pena di provare. LXP

VERDETTO

SVILUPPATORE: Longterm Games S.A.
WEB: https://spacetail.com
PREZZO: 16,79 €

GIOCABILITÀ	8/10	LONGEVITÀ	8/10
GRAFICA	8/10	QUALITÀ/PREZZO	8/10

Un gioco particolare e coinvolgente, con una bella grafica e una storia accattivante. Vi consente di sfoggiare abilità e ingegno da una prospettiva nuova.

» Il voto di Linux Pro 8/10

OpenMV Cam H7 R2

Creare un robot in grado di vedere diventa facile con questa scheda

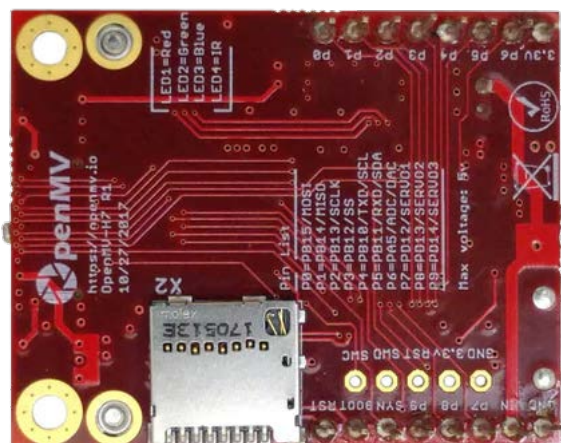
SPECIFICHE

Processore: STM32H743VI ARM Cortex M7 a 480 MH
Memoria: 1 MB di SRAM e 2 MB di flash
Connettori: USB (12 Mbs), slot per schede µSD, bus SPI, bus I²C, bus CAN, bus seriale asincrono, ADC a 12 bit, DAC a 12 bit, tre pin di I/O per i servo

Creare un modulo di **visione artificiale** piccolo, economico ed espandibile per avvicinare gli algoritmi di **computer vision** a **maker** e **hobbisti**: questo è l'obiettivo del progetto **OpenMV**, che dichiara di voler diventare "l'**Arduino** della visione artificiale". **OpenMV Cam H7 R2** è una videocamera montata su un **microcontrollore** con **MicroPython**. Questo approccio consente di scrivere **script Python** da eseguire su OpenMV, che si compilano e si caricano facilmente, e offre i vantaggi tipici dei microcontrollori, come il basso consumo energetico e l'elaborazione istantanea per le applicazioni **embedded**. L'esecuzione di algoritmi di visione artificiale su ciò che la telecamera OpenMV vede è semplificata e consente di rilevare rapidamente elementi come colori e volti e utilizzarli per controllare i **pin di I/O**.

Versatilità a tutto tondo

Oltre a salvare immagini in **scala di grigi** o **RGB565** (in formato **BMP**, **JPG**, **PPM** e **PGM**) e video **MJPEG** su una scheda **µSD** collegata, il modulo dispone di algoritmi integrati in grado, per esempio, di rilevare volti, identificare gli occhi e tracciare le pupille. Offre il supporto per **TensorFlow** e il produttore mostra come addestrare un **dataset** di riconoscimento dei sorrisi in meno di **30 minuti** in un video su **YouTube** (**https://bit.ly/3WWW6d3**). OpenMV Cam viene fornita con una libreria **RPC** (**Remote Python/Procedure Call**) per collegarla al PC, a un **Single Board Computer** come la **RaspberryPi** o a un microcontrollore come **Arduino**. La libreria funziona su **bus seriali asincroni** (**UART**), **I²C**, **SPI** e **CAN**, tramite la porta **Virtual COM USB** (**VCP**) e su **Wi-Fi** utilizzando il **Wi-Fi Shield**. Infatti la sua disposizione standard dei pin di I/O consente di impilare gli **shield** su di essa come su una scheda Arduino. Sono disponibili uno shield di **prototipazione** per creare il proprio circuito personalizzato, uno per la **termografia** per vedere al buio e uno **LCD**. La telecamera viene fornita con un obiettivo da **2,8 mm** ma ha un attacco standard **M12** che consente di cambiarlo in base alle esigenze del progetto. OpenMV vende uno **zoom telescopico 4X**, un **fish-eye ultra-grandangolare a 185°** e un obiettivo senza **filtro IR** per l'uso con applicazioni di tracciamento a infrarossi, ma potete acquistare molti altri obiettivi M12 da altri fornitori. La possibilità di programmare OpenMV Cam utilizzando **Python 3** e le sue librerie rende molto più semplice la creazione di progetti con algoritmi di visione artificiale. OpenMV ha anche un proprio **IDE** (**https://openmv.io/pages/download**) disponibile per **Ubuntu** a **32** e **64 bit** e per Raspberry Pi. La documentazione è valida e dettagliata, anche se in inglese. LXP

Un **micro controllore** con videocamera programmabile in **Python** per utilizzare la visione artificiale in modo semplice

VERDETTO

PRODUTTORE: OpenMV
WEB: https://openmv.io/
PREZZO: 82 €

CARATTERISTICHE	9/10	FACILITÀ D'USO	9/10
PRESTAZIONI	9/10	QUALITÀ/PREZZO	9/10

Un ottimo prodotto per dare ai vostri progetti e ai vostri robot il dono della vista in modo veloce e con molte possibilità creative.

» **Il voto di Linux Pro** 9/10

Wacom One 13,33”

Una tavoletta grafica valida, utilizzabile con PC, Chromebook e cellulare

SPECIFICHE

Dimensioni display: 33,8 cm (13,3”)
Risoluzione: Full HD 1.920 x 1.080
Dimensioni: 225 x 357 x 14,6 mm
Peso: 1,0 kg

Con la **Wacom One**, il noto produttore giapponese di tavolette grafiche propone un prodotto versatile, con un buon rapporto qualità/prezzo, perfetto per gli studenti e per chi disegna o fa lavori grafici per passione. Il **display** da **13,3”** ha quasi le dimensioni di un foglio **A4** e offre un **attrito superficiale** naturale progettato per replicare l’esperienza della scrittura a mano o del disegno su carta. Inclusa con la tavoletta c’è la **Penna Wacom One** sensibile alla pressione, senza cavi e senza batteria (assorbe l’alimentazione dal display attraverso la tecnologia di **risonanza elettromagnetica EMR** della casa). Con i suoi **4.096** livelli di pressione e un pulsante laterale personalizzabile, nell’uso risulta responsiva e offre un’esperienza naturale di scrittura e disegno sulla tavoletta. Non offre l’assoluta precisione della **Wacom Pro Pen 2** ma, a meno che non siate dei professionisti della grafica, risponderà a tutte le vostre esigenze, che si tratti di prendere appunti, creare schizzi o disegnare il vostro fumetto personale con un programma grafico.

Software e compatibilità

Con l’acquisto della tavoletta ottenete anche diverse licenze (fino a sei mesi) per noti software creativi come **Clip Studio Paint**, **Photo 2** e **Designer 2** di **Affinity** e altri ancora. Coprono varie attività, da disegno e pittura al fotoritocco e alla grafica vettoriale. Trovate un elenco del software **in bundle** all’indirizzo **https://bit.ly/3X5CKCt**. La Wacom One si collega al computer con le connessioni **HDMI** e **USB-A** e la confezione include il suo alimentatore. Anche se le specifiche della tavoletta la vedono compatibile solo con **Windows** e **macOS**, Wacom contribuisce ai **driver** del **kernel Linux** dal **2002** e oggi tutte le funzionalità hardware dei suoi prodotti sono supportate dall’**OS** di **Tux**, anche se alcune applicazioni software potrebbero non avere il supporto completo. Molte distribuzioni hanno il driver preinstallato, ma nel caso la vostra non lo includesse potete collegarvi a **https://linuxwacom.github.io/** per scaricare quel che vi serve. La tavoletta è inoltre compatibile con diversi **Chromebook** e, con un adattatore supplementare, a una serie di dispositivi **Android**, soprattutto di **Samsung** e **Huawei**. Trovate un elenco dell’hardware supportato all’indirizzo **https://www.wacom.com/it-it/comp**. Nel complesso è un ottimo prodotto per avvicinarsi alle tavolette grafiche: imparare a usarla è facile, il funzionamento è fluido e potete collegarla sia al computer sia a un dispositivo mobile. LXP

È compatibile con diversi **Chromebook** e (con un adattatore) con vari dispositivi **Android**

VERDETTO

PRODUTTORE: Wacom
WEB: www.wacom.com/it-it
PREZZO: 409,90 €

CARATTERISTICHE	8/10	FACILITÀ D’USO	9/10
PRESTAZIONI	8/10	QUALITÀ/PREZZO	8/10

Un prodotto solido e affidabile che non si rivolge al mercato professionale ma offre comunque ottime prestazioni e un’apprezzabile versatilità.

» Il voto di Linux Pro 8/10

Puppy Linux 9.5

Distribuzioni piccole e scattanti che vi permettono di avere tutto ciò che serve in poco spazio e di riportare in uso vecchie macchine

IN BREVE

Una famiglia di distribuzioni molto piccole ma ricche di funzioni. Puppy Linux si avvia nella RAM, quindi tutte le applicazioni partono in un batter d'occhio e rispondono subito agli input dell'utente. Facile da personalizzare e utilizzare anche con macchine molto vecchie.

Puppy Linux non è una distribuzione ma una famiglia di **distro Linux** destinate agli utenti domestici. Si basano sugli stessi principi condivisi, utilizzano il medesimo set di strumenti e sfruttano un parco di applicazioni e configurazioni specifiche per i "puppy" (che in inglese vuol dire cucciolo). In generale, forniscono comportamenti e funzionalità coerenti, a prescindere dalla versione scelta. Il progetto è stato originariamente creato da **Barry Kauler** nel **2003** come versione più snella ma al contempo altrettanto completa di **Vector Linux**. Un altro obiettivo principale è sempre stato quello di offrire distro utili fin dall'inizio: la **ISO** che si scarica contiene tutte le applicazioni standard per gestire le esigenze informatiche più comuni, come un elaboratore di testi, un foglio di calcolo e un browser Web.

Puppy offre una famiglia di distribuzioni leggere, veloci e personalizzabili con un ampio parco di applicazioni specifiche e anche la possibilità di usare quelle di **Ubuntu**

Una famiglia variegata

Un'altra idea centrale del progetto è che il sistema operativo funzioni come un **Live-CD** ma con il supporto per la "persistenza", ovvero che mantenga i dati tra le sessioni. In questo modo, il Live-CD può essere utilizzato come se fosse un sistema operativo installato. In caso di problemi, la persistenza può essere disattivata e Puppy si riavvia nel suo stato incontaminato. Anche le dimensioni sono sempre state una considerazione importante, con la scelta di componenti software che forniscono la massima funzionalità con ingombri modesti, come per esempio il file manager **ROX Filer**. Questo è solitamente combinato con il gestore di finestre leggero **JWM** (**http://joewing.net/projects/jwm/**) e le distro Puppy Linux sono in genere di **300 MB** o meno. Un'altra caratteristica comune a questa famiglia di sistemi operativi è la facilità d'uso: si dichiarano "certificati a misura di nonno" e si impegnano per essere semplici da usare per tutti. Le distribuzioni rientrano in tre grandi categorie. In primo luogo ci sono quelle ufficiali mantenute dal team di Puppy Linux, di solito destinate a scopi generici e create con la sua raccolta di **script per la compilazione**, **Woof-CE** (**https://puppylinux-woof-ce.github.io/woof-ce.html**). Poi ci sono le distribuzioni che utilizzano Woof-CE con alcuni pacchetti aggiuntivi o modificati, spesso mirate a soddisfare esigenze specifiche. Infine, ci sono le derivate non ufficiali ("**puplet**") create e mantenute dagli appassionati di Puppy Linux, di solito mirate a scopi precisi. Considerando tutte le tipologie, ci sono versioni davvero per tutti i gusti e, soprattutto se avete una macchina con risorse limitate, sono sicuramente una soluzione da provare. Puppy offre molte applicazioni specifiche ed è anche compatibile con i repository di **Ubuntu Focal Fossa**. La sua struttura modulare consente di sostituire il **kernel**, le applicazioni e il **firmware** molto rapidamente e di avere subito una distro snella, veloce e funzionale su misura. LXP

SPECIFICHE

Minime
Memoria: 300 MB
CPU: Pentium 900 MHz
Build: x86 64 bit, x86 32 bit ARM

VERDETTO

PRODUTTORE: Il team di Puppy Linux
WEB: puppylinux-woof-ce.github.io/index.html
LICENZE: Varie

CARATTERISTICHE	9/10	FACILITÀ D'USO	8/10
PRESTAZIONI	8/10	DOCUMENTAZIONE	7/10

Tante funzionalità e possibilità di personalizzazione in poco spazio per un progetto che ha qualcosa per tutti.

» Il voto di Linux Pro 8/10

Salix 15.0

Nonostante la procedura di installazione non proprio intuitiva, l'uso di questa leggerissima distro risulta facile anche per i neofiti

IN BREVE

Una distro per desktop basata su Slackware, disponibile sia per computer con architettura a 32 bit, sia per quelli con architettura a 64 bit. L'ambiente desktop è il leggerissimo Xfce che ne permette il funzionamento anche sulle macchine più obsolete.

SPECIFICHE

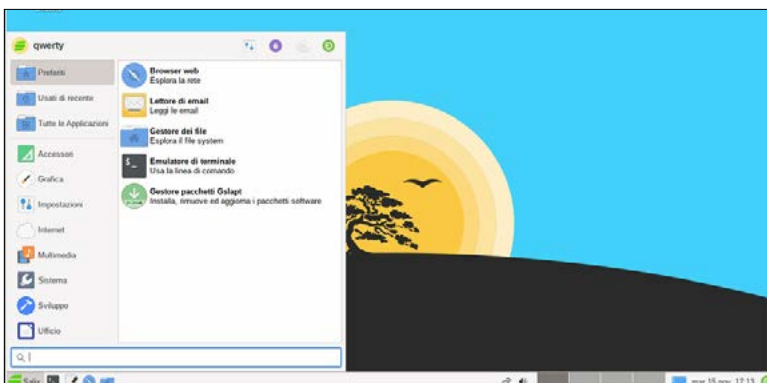
CPU: 1 GHz
Memoria: 512 MB
HDD: 8 GB
Build: 32 e 64 bit

Non lasciatevi spaventare dalla procedura di installazione! L'aspetto è minimalista e sembra di essere tornati indietro nel tempo di almeno quindici anni. Inoltre, non è intuitiva come quella di molte altre distribuzioni Linux.
Per fortuna, però, c'è una comoda documentazione in italiano, particolarmente ben fatta, che troverete collegandovi all'indirizzo **https://tinyurl.com/ph68e7dt**. Grazie a essa dovreste superare agevolmente questo scoglio, anche se non siete particolarmente esperti del mondo Linux.
Comunque, una volta installato, Salix non presenta più alcun particolare problema. La sua interfaccia risulta piuttosto facile da usare e l'ambiente desktop **Xfce** la rende particolarmente agile e reattiva. Sempre parlando di installazione, avrete la possibilità di scegliere tra tre tipi: **Full**, **Basic** e **Core**. La prima è ovviamente l'installazione completa, mentre la seconda è quella minima. La terza è più particolare perché non comprende l'ambiente grafico ed è pensata per essere usata come versione **server**.

La procedura di installazione di **Salix** ha un aspetto estremamente spartano (soprattutto se paragonata a quella di altre distribuzioni Linux) che potrebbe spaventare i meno esperti

Muoversi dentro Salix

Il desktop di questo sistema operativo mostra cinque icone: **Cestino**, **File system**, la cartella **Home**, **Salix Online** (che è il collegamento alla sua pagina principale tramite **Firefox**) e **Salix IRC support**. In basso c'è la classica barra che racchiude tutto ciò che vi serve, dal menu principale, al **Terminale**, al selettore degli spazi di lavoro, fino al pulsante di spegnimento, che ritrovate in alto a destra proprio nel menu principale. Quest'ultimo è organizzato in modo semplice: a sinistra ci sono le cartelle e le aree tematiche e a destra c'è il loro contenuto, quindi è impossibile perdersi. La dotazione delle applicazioni è molto ricca e può riservare qualche sorpresa. Per esempio, nella sezione **Ufficio** vedrete che la suite di **LibreOffice** è completa anche di **Base**, che normalmente in altri sistemi operativi va installato a parte. Tra gli altri software disponibili troverete **GIMP**, **Pidgin**, **Parole** come riproduttore di file multimediali e uno strumento per installarne i **codec**. Salix è già compatibile con le applicazioni in formato **Flathub** e nella sezione **Sistema** c'è il collegamento diretto alla pagina ufficiale del loro archivio online. Gli sviluppatori avranno già a disposizione **Geany** e **Meld**. LXP

Pur essendo molto leggero, **Salix** è comunque ricco di strumenti per ogni necessità

VERDETTO

PRODUTTORE: Il team di sviluppo di Salix
WEB: https://www.salixos.org
LICENZE: GPL e altre

CARATTERISTICHE	8/10	FACILITÀ D'USO	9/10
PRESTAZIONI	9/10	DOCUMENTAZIONE	10/10

Una distro che si autodefinisce bonsai, leggera e adatta a qualsiasi tipo di macchina, ma ricca di risorse.

» Il voto di Linux Pro 9/10

Da non perdere

Vup » HomeBank » MusicBrainz Picard » SoapUI » Mepo » SMPlayer » AES Crypt » WhatSie » jExifToolGUI » Syncthing » Speed Dreams » Bitfighter

CLOUD DECENTRALIZZATO

Vup

Versione: 0.12.BETA
Web: https://vup.app/

Con **Vup** potete salvare i vostri file sul **cloud** senza dipendere da giganti informatici e sfruttando un protocollo **decentralizzato**

Vup è un'applicazione di archiviazione sul **cloud** privata e decentralizzata basata sul protocollo aperto per l'**hosting** di dati e applicazioni Web **Skynet** (**https://github.com/SkynetLabs**). Il sistema è mirato alla tutela della vostra privacy e tutto è crittografato **end-to-end**, compresi tutti i **metadati**. Vup utilizza persino tecniche avanzate come il **padding** per nascondere le dimensioni reali dei file. Skynet offre una delle reti di **storage** decentralizzato più veloci e reattive e con Vup è possibile utilizzare tutti i suoi portali o anche gestirne uno proprio, con in più l'opzione di cambiare portale in qualsiasi momento senza perdere nulla. Potete condividere file o **directory** anche con chi non ha installato Vup. Quest'ultimo è scritto in **Flutter** ed è compilato in **codice nativo**, il che comporta prestazioni più elevate e un minore utilizzo di memoria rispetto alle app basate su **Electron**. Il programma è ancora in fase **Beta** ma è molto promettente e vale la pena di provarlo. Per farlo andate su **https://github.com/redsolver/vup/releases/latest** e scaricate l'ultimo file **...-BETA-Linux.AppImage**. Fate click con il tasto destro del mouse sul file scaricato nell'**esploratore di file** e rendetelo eseguibile o scrivete nel terminale:

```
chmod +x ...-BETA-Linux.AppImage
```

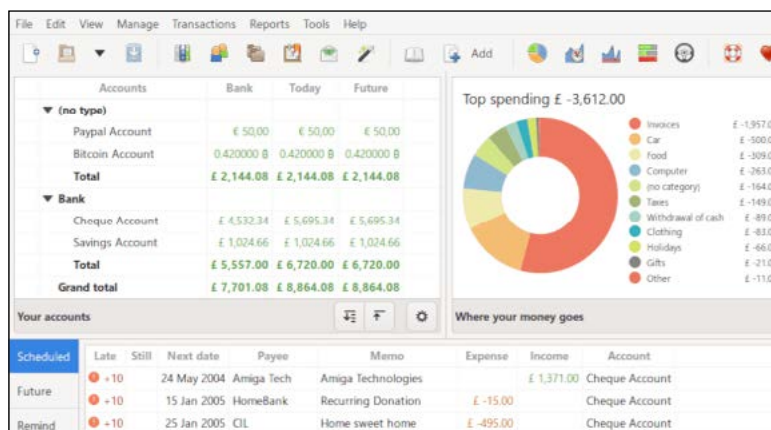
Fate doppio click sul file AppImage per avviarlo.

GESTIONE FINANZE

HomeBank

Versione: 5.5.7 **Web:** http://homebank.free.fr/en/downloads.php

Tener traccia delle spese sul conto della banca, di **PayPay** e anche di eventuali **wallet di criptovalute** diventa molto semplice

HomeBank è un software maturo, con oltre **24** anni di storia e **feedback** da parte degli utenti. È progettato per analizzare in modo relativamente semplice le finanze personali e il bilancio nel dettaglio, utilizzando potenti strumenti di filtraggio e grafici di facile lettura. Può essere utile sia per una piccola impresa sia per l'amministrazione domestica. Aiuta a prevedere i costi futuri inserendo transazioni programmate e offre strumenti di **reportistica** dinamici e potenti. Permette di importare gli estratti conto scaricati dalla banca, dal sito della carta di credito (o magari esportati da altri programmi) nei formati **OFX**, **QFX** e **QIF** e include funzioni di rilevamento dei duplicati. Tra le sue numerose funzionalità, **Budget** è particolarmente utile in quanto consente di monitorare le spese per categoria e sottocategoria in rapporto a un budget preimpostato. HomeBank è disponibile per **GNU/Linux**, **FreeBSD** e **Windows** e anche (con **porting** di terze parti) su **Android**, **iOS** e **Nokia N**.
Lo trovate già incluso nelle principali distribuzioni e per installarlo in **Debian** e **Ubuntu** basta usare:

```
'apt-get install homebank'
```

TAG PER FILE AUDIO

MusicBrainz Picard

Versione: 2.8.3
https://picard.musicbrainz.org/

Un **database** di **metadati** musicali e un'applicazione di gestione dei **tag** per ottenere il massimo dalla vostra collezione

MusicBrainz Picard è un sistema multipiattaforma per **taggare** i file musicali. Quelli che si scaricano tipicamente contengono già alcuni **metadati** di base, ma ci sono letteralmente centinaia di **tag** che possono essere applicati alla vostra musica per avere una collezione più organizzata e ricca. Invece di cercare tutte queste informazioni per ogni album e brano singolarmente e inserirle in uno strumento di **tagging**, è più facile e veloce ottenere i dati da un **database** condiviso accessibile all'applicazione di gestione dei tag. Questo compito è svolto da MusicBrainz Picard. MusicBrainz è il database e Picard è lo strumento che **tagga** i file musicali. Quest'ultimo utilizza le impronte digitali audio **AcoustID** (che consentono di identificare i file in base alla musica anche se non hanno metadati) e può esaminare interi **CD** con un click. Fornisce innumerevoli personalizzazioni per adattare le raccolte musicali alle vostre esigenze, ma se avete bisogno di una particolare funzione che non trovate potete scegliere tra una selezione di **plug-in** disponibili (**https://picard.musicbrainz.org/plugins/**) o scriverne di vostri. Il progetto ha una comunità attiva (**https://community.metabrainz.org**/) ed è ben documentato (**https://picard-docs.musicbrainz.org/en/index.html**). Da provare.

TEST API E SERVIZI WEB

SoapUI

Versione: 5.7.0
Web: https://www.soapui.org/

SoapUI copre l'intero spettro dei test per **API** e servizi Web: funzionali, di sicurezza, di carico e di simulazione

Le **API** (**Application Programming Interfaces**) sono diventate un elemento centrale dello sviluppo del software, in quanto collegano e trasferiscono dati e logica tra sistemi e applicazioni diversi. Come per ogni cosa, se le sviluppate è fondamentale eseguire dei buoni test. **SoapUI Open** copre l'intero spettro dei test: **funzionali**, **di sicurezza**, **di carico** e **di simulazione (mocking)**. Consente di verificare con facilità i servizi Web basati su **REST**, **SOAP** e **GraphQL** utilizzando un'interfaccia grafica. Permette di eseguire test funzionali senza scrivere codice creandoli con il **drag and drop** e di strutturare test di carico in modo rapido e semplice sulla base dei test funzionali di API esistenti. È difficile sopravvalutare i vantaggi dell'esecuzione di test funzionali prima di passare al controllo qualità. Tiene traccia di come si comporta il vostro prodotto e senza di esso il vostro software rischia di bloccarsi quando gli utenti crescono. Con questo strumento diventa più semplice farli, abbattendo tempi e costi. Anche l'**API Mocking** è molto importante, perché consente di simulare i servizi Web reali senza dover aspettare che siano pronti o accessibili. Elimina la necessità di costruire costose repliche in scala reale del sistema di produzione, creando un servizio virtuale che funziona come quello autentico. Un servizio ricreato con il mocking imita una vera **API REST 4** o SOAP: contiene le definizioni delle operazioni che i **client** richiamano, riceve le richieste e restituisce risposte simulate. Potete così prepararvi meglio al lancio del prodotto. SoapUI offre anche test di sicurezza tramite una serie di prove e scansioni per proteggere i vostri servizi sui siti Web dalle vulnerabilità più comuni e rappresenta uno strumento completo ed efficace che può facilitare la vita a tutti gli sviluppatori.

VISUALIZZATORE DI MAPPE

Mepo

Versione: 1.0
Web: http://mepo.milesalan.com/

Mepo è un visualizzatore di mappe **OpenStreetMap** (**OSM**), il noto progetto collaborativo finalizzato a creare mappe del mondo a contenuto libero. È veloce, facile da usare, modificabile ed è disponibile per desktop con **Linux** e per dispositivi mobili basati su questo sistema operativo (come **PinePhone**, **Librem 5**, **pmOS**, ecc.). Entrambi gli ambienti hanno diverse interfacce utente e utilizzano sia **Wayland** sia **X**. Mepo funziona sia **offline** sia **online** e la sua interfaccia utente è progettata (abbracciando la classica filosofia **UNIX**) per concentrarsi su un singolo compito e svolgerlo al meglio. Questo compito è scaricare e **renderizzare** mappe attraverso un'interfaccia minimalista controllabile sia con il tocco o il mouse, sia con la tastiera. Le funzionalità aggiuntive sono attivate tramite il suo **linguaggio di comando/API** (**Mepolang**) in grado di essere **scriptato** per fornire elementi quali **script di ricerca** personalizzati, l'individuazione della posizione, la creazione di segnalibri e molto altro. Questo riduce la complessità della logica complessiva dell'applicazione e la velocizza. Mepo è compatibile con diversi ambienti mobili Linux, tra cui **Phosh**, **Sxmo** e **Swmo**. La versione uscita a ottobre ha apportato miglioramenti alla documentazione e aggiunto il supporto per **Plasma Mobile**. Per installarlo come **flatpak** usate

```
flatpak install flathub com.milesalan.mepo
```

mentre per eseguirlo serve il comando

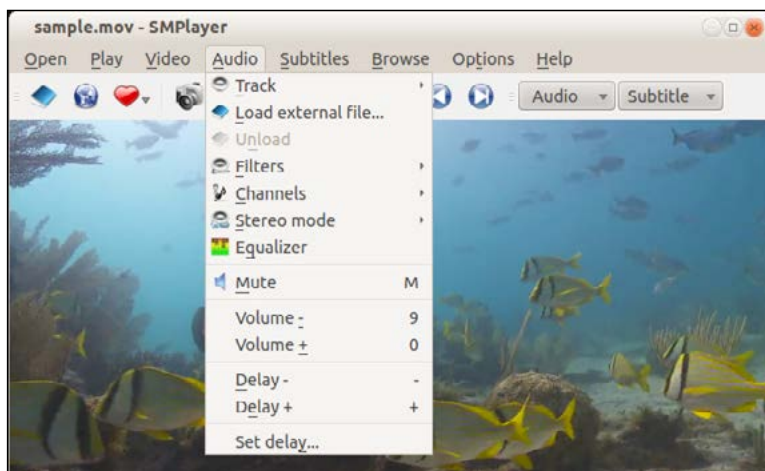
```
flatpak run com.milesalan.mepo
```

Un visualizzatore di mappe **OpenStreetMap** (**OSM**) che funziona anche su dispositivi mobili basati su **Linux**

RIPRODUTTORE MULTIMEDIALE

SMPlayer

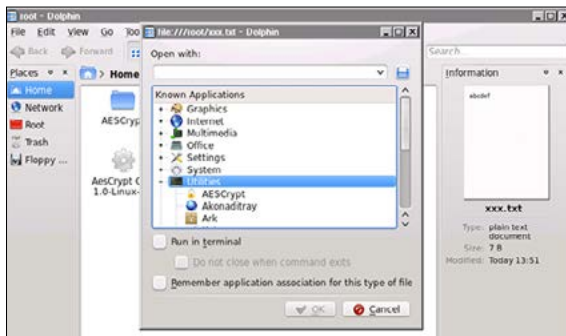
Versione: 22.7.0 **Web:** https://www.smplayer.info/en/info

SMPlayer è un lettore multimediale con interfaccia grafica basato su **MPlayer**. I suoi **codec** integrati consentono di riprodurre praticamente tutti i formati video e audio. Inoltre, l'applicazione ricorda le impostazioni di tutti i file riprodotti, per esempio la traccia audio, i sottotitoli e il volume di un film. In questo modo, se si deve interrompere la visione, non è necessario impostarle di nuovo. SMPlayer può riprodurre i filmati di **YouTube** e anche sfogliarli e cercarli tramite l'applicazione **SMTube**. Il fatto di lanciare i video con un riproduttore multimediale anziché con un lettore **Flash** consente di ottenere prestazioni migliori, in particolare con i contenuti **HD**. Una caratteristica utile di SMPlayer è che può cercare e scaricare i sottotitoli da **opensubtitles.org**. Basta infatti aprire un video e selezionare **Find subtitles on opensubtitles.org** nel menu **Subtitles**. Appare quindi una nuova finestra con un elenco di quelli disponibili. Selezionate il sottotitolo che desiderate scaricare e fate click sul pulsante **Download** per visualizzarlo automaticamente nel video. Ci è piaciuta anche l'opzione di riproduzione su **Chromecast**. Basta selezionare **Play on Chromecast** dal menu **Play** per aprire la pagina di controllo di **SMPlayer Chromecast** nel browser e comunicare da lì con il dispositivo Chromecast. Avete le opzioni per connettervi, disconnettervi, avviare la riproduzione e controllarla, cosa che potete anche fare con uno **smartphone** o un **tablet** scansionando il **codice QR** nella pagina. È anche possibile inviare il video a un secondo **display** collegato al computer mentre si controlla l'applicazione sul **monitor** principale.

Un lettore multimediale facile da usare con **codec** integrati e una serie di utili funzioni

CRIPTAZIONE

AES Crypt

Versione: 3.10
Web: https://www.aescrypt.com/

AES Crypt è un software di **criptazione** dei file che utilizza l'algoritmo **Advanced Encryption Standard** (**AES**), ossia quello usato come standard dal governo **USA** per la sua potenza. Il programma mira a rendere il suo utilizzo facile. Per **crittografare** un file, basta fare click con il tasto destro o sinistro (a seconda del desktop) su di esso e aprirlo con AES Crypt. A questo punto vi viene richiesto di inserire la password desiderata e l'applicazione produce un file che non può essere letto da nessuno che non la conosca. Per eseguire la stessa operazione dal **terminale** è sufficiente inserire il comando **aescrypt** con gli **argomenti** della riga di comando appropriati. Per esempio, supponiamo di avere un file chiamato **contratto.jpg** da crittografare con la password **32124G**. Si immette il seguente comando:

Con oltre un milione di scaricamenti, **AES Crypt** è un metodo semplice e popolare per **criptare** i file

```
aescrypt -e -p 32124G contratto.jpg
```

Questo è quanto! Il programma crea un file con il nome **contratto.jpg.aes**. Quando volete decriptarlo, se usate l'interfaccia grafica dovete semplicemente fare doppio click sul file **.aes** e inserire la vostra password segreta quando vi viene richiesta. Dal **terminale** si deve invece immettere il seguente comando:

```
aescrypt -d -p 32124G contratto.jpg.aes
```

Se usate AES Crypt per decriptare un file in un ufficio con altre persone e non volete mostrare la password sulla riga di comando, basta escludere il **parametro -p** in questo modo:

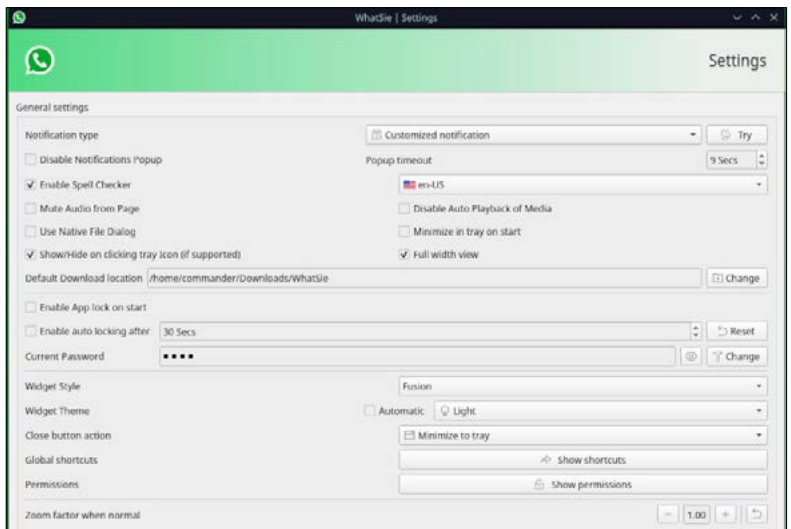
```
aescrypt -d contratto.jpg.aes
```

Il programma chiederà la password ma quando la inserirete non verrà visualizzata sullo schermo.

CLIENT WHATSAPP

WhatSie

Versione: v4.10.2 **Web:** https://github.com/keshavbhatt/whatsie

WhatSie è un **client Web** per **WhatsApp** per computer desktop in ambiente **Linux** basato su **Qt WebEngine**. Ricco di funzioni e facile da usare, offre numerose impostazioni che consentono di personalizzare l'uso dell'applicazione sul PC. Se, per esempio, avete bisogno di un po' di privacy, potete attivare la modalità **Do not disturb** o mettere in **Mute** tutto l'audio da WhatsApp. Se volete vedere qualcosa in modo più chiaro c'è la modalità **Full view** che consente di espandere la vista principale all'intera larghezza della finestra. È inoltre possibile disattivare la riproduzione automatica dei file multimediali e usare le scorciatoie da tastiera per controllare l'app. Sono disponibili anche temi chiari e scuri con la possibilità di cambiarli in modo automatico di giorno e di sera. L'app si può anche controllare dalla riga di comando, con una serie di opzioni che consentono di interagire con le istanze già in esecuzione di WhatSie. Questo è un elenco riassuntivo dei flag più utili:

```
-h, --help  Guida alle opzioni da riga di comando
 -v, --version       informazioni sulla versione
 -w, --show-window   Mostra la finestra principale
dell'istanza in esecuzione di WhatSie
 -s, --open-settings  Apre la finestra di dialogo
delle impostazioni in un'istanza in esecuzione di
WhatSie
 -t, --toggle-theme   Passa dal tema scuro a quello
chiaro e viceversa
 -r, --reload-app     Ricarica l'app
 -n, --new-chat       Apre un nuovo prompt di chat
in un'istanza in esecuzione di WhatSie
```

Mille opzioni di personalizzazione e controllo dalla **riga di comando** per un modo diverso di usare **WhatsApp**

GESTIONE METADATI

jExifToolGUI

Versione: 2.0.1 **Web:** https://github.com/hvdwolf/jExifToolGUI

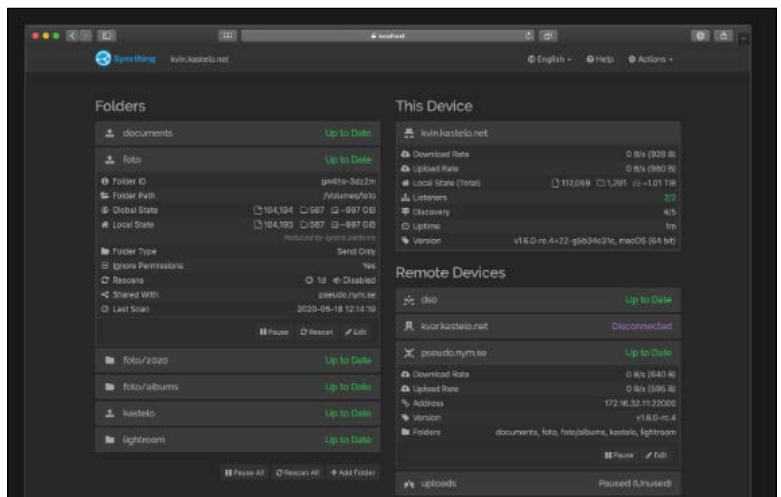
Con **jExifToolGUI** potete leggere e scrivere i **metadati** da e sui file, in particolare quelli di immagini. È fondamentalmente un **front-end** grafico per **ExifTool** (**https://exiftool.org/**), un'applicazione a riga di comando **Perl** indipendente dalla piattaforma che è anche una libreria per leggere, scrivere e modificare le **meta informazioni** in un'ampia varietà di file. ExifTool è più completo ma molti trovano che lavorare con le immagini sia più facile con un'interfaccia grafica. jExifToolGUI dispone di alcune schermate preformattate per i **tag EXIF**, **gps**, **xmp**, **gpano** e per un set molto limitato di **IPTC** e supporta anche il **geotagging**. Inoltre è possibile definire la propria combinazione di tag di metadati da scrivere sulle immagini utilizzando qualsiasi tag supportato da ExifTool. Potete anche definire "nuovi" tag non esistenti da aggiungere ai vostri brani con un file di configurazione personalizzato. Il programma offre numerose funzionalità e una notevole flessibilità per quanto riguarda le modalità di lettura/scrittura di dati da e verso le immagini. Nel caso in cui ciò non fosse sufficiente è anche possibile creare i propri comandi ed eseguirli sulle immagini. Potete quindi salvarli come preferiti per poterli riutilizzare facilmente in un secondo momento. Nel complesso, jExifToolGUI facilita notevolmente il lavoro con i tag dei file immagine e vale la pena di provarlo se non si trova comoda la riga di comando.

La semplice interfaccia consiste di un pannello di sinistra con le foto e uno di destra composto da una serie di schede con le azioni disponibili

SINCRONIZZAZIONE FILE

Syncthing

Versione: v1.22.0
Web: https://syncthing.net/

Syncthing è un programma di sincronizzazione continua dei file tra due o più computer in tempo reale. È un'ottima opzione per tutelare privacy e riservatezza, dato che i vostri dati non sono mai memorizzati in nessun altro luogo che non sia il vostro computer. Non esiste un server centrale che possa essere compromesso, in modo legale o illegale. Tutte le comunicazioni sono criptate e protette utilizzando il protocollo **Transport Layer Security** (**TLS**) che tutela la comunicazione dalla sorgente al destinatario (**end-to-end**) sia per l'integrità dei dati sia per la confidenzialità. Syncthing presta anche molta attenzione all'autenticazione. Ogni dispositivo è infatti identificato da un solido **certificato crittografico** e solo quelli che avete autorizzato esplicitamente possono connettersi alle vostre altre macchine. Un altro punto di forza di Syncthing è la semplicità. La configurazione e il monitoraggio si fanno attraverso un'interfaccia chiara e accessibile tramite browser. Il programma non ha inoltre bisogno di **indirizzi IP** o di configurazioni avanzate per funzionare su **LAN** e **Internet**. Ogni macchina è identificata da un **ID**, che date ai vostri contatti per condividere le cartelle: **UPnP** è sufficiente se non volete o non sapete fare il **port forwarding**. Potete sincronizzare tutte le cartelle di cui avete bisogno con diverse persone o solo tra i vostri dispositivi ed eseguire Syncthing sui computer desktop sincronizzandoli con un server per il **backup**. Il progetto ha una **community** attiva (**https://forum.syncthing.net/**) e una documentazione dettagliata su **https://docs.syncthing.net/**.

Un'ottima opzione per condividere e sincronizzare le proprie cartelle tutelando privacy e riservatezza

SIMULATORE AUTOMOBILISTICO

Speed Dreams

Versione: 0.32.0 **Web:** https://www.speed-dreams.net/en/

Una comunità di sviluppo reattiva e aperta ai suggerimenti, bella grafica e realismo fisico sono tra i principali pregi di **Speed Dreams**

Speed Dreams è un simulatore di corse automobilistiche che punta al massimo realismo attraverso una grafica **3D** di buona qualità e un motore fisico accurato. È nato come **fork** di **TORCS** (**The Open Racing Car Simulator**, **https://torcs.sourceforge.net/**), una simulazione di gare automobilistiche multipiattaforma utilizzata come normale sfida di abilità, come gioco di corse basato sull'Intelligenza Artificiale e come piattaforma di ricerca. Con il tempo, però, Speed Dreams ha ampliato ulteriormente il realismo visivo e la simulazione fisica, grazie a un team di sviluppo attivo e a una comunità in crescita. L'obiettivo principale del gruppo è quello di implementare nuove caratteristiche, auto, tracciati e avversari controllati dall'intelligenza artificiale per rendere il titolo più divertente per il giocatore, spingendo costantemente verso il realismo visivo e fisico. Speed Dreams è anche pensato per l'uso in attività di ricerca, studio o insegnamento grazie alla sua licenza **GPL V2+** e all'architettura chiara e modulare della sua base di codice in **C/C++**. La comunità incoraggia la partecipazione e gli sviluppatori possono provare le loro idee e avere ottime possibilità di vederle pubblicate per gli utenti finali. Questi ultimi possono godersi il frutto di queste idee e dare la propria opinione in merito o dare nuovi suggerimenti. Se pensate che le vostre proposte di **patch** per TORCS non vengano integrate nella versione ufficiale con la rapidità che desiderate, potete proporle qui.

ARCADE MULTIGIOCATORE

Bitfighter

Versione: 022
Web: http://bitfighter.org/

Combattimento spaziale a squadre per più giocatori con mappe create dagli utenti e tutto il fascino di un **arcade** in stile **vintage**!

Bitfighter è un titolo di combattimento spaziale a squadre per più giocatori veloce e divertente. Ci sono molte modalità di gioco e mappe uniche. Gli utenti possono inoltre creare le proprie utilizzando l'**editor** integrato e giocare a livelli realizzati da altri.
In Bitfighter si controlla una navicella triangolare e l'obiettivo dipende dal tipo di gioco selezionato. Potreste dover tentare di catturare una bandiera, tirare una palla in porta, raccogliere bandierine o catturare zone. Entrerete in contatto con giocatori nemici che dovrete combattere con le varie armi che avete scelto, ognuna delle quali ha un proprio scopo strategico specifico. Il **Bouncer**, per esempio, è un tipo di proiettile che rimbalza sulle pareti, molto utile per un livello con corridoi stretti. È inoltre possibile scegliere e utilizzare una serie di moduli, ognuno dei quali ha un determinato scopo. I giocatori nemici non sono, inoltre, il vostro unico ostacolo. Ci sono anche torrette stazionarie che sparano in automatico e sono fortemente determinate a eliminarvi. I raggi laser nemici non vi danneggiano ma cercano di bloccarvi il cammino. Fortunatamente, entrambi i dispositivi possono essere distrutti, anche se farlo non è facile. Uno dei moduli menzionati è **Repair**, che permette di ripristinare non solo se stessi ma anche i compagni di squadra, le torrette neutrali e i raggi laser per infastidire gli avversari. Questi sono solo alcuni esempi delle opzioni di gioco e degli oggetti integrabili di cui Bitfighter è ricco, come asteroidi, teletrasporti e altro ancora. Le sue opzioni di personalizzazione e l'interazione con la **community** lo rendono molto divertente e la sua grafica non mancherà di incantare i nostalgici. LXP

Tante soluzioni provate per voi per farvi scegliere sempre il meglio sul mercato

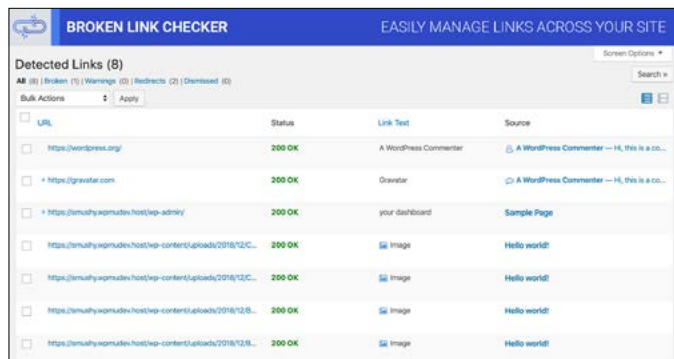
Plug-in per WordPress

Qualsiasi funzione vogliate integrare nel vostro sito, dalla vendita all'ottimizzazione SEO, può essere facilitata da un add-on: ecco i migliori!

I CRITERI

In questa selezione abbiamo raccolto alcuni degli strumenti più pratici e potenti per ottenere il massimo dai vostri siti in WordPress sotto ogni profilo.

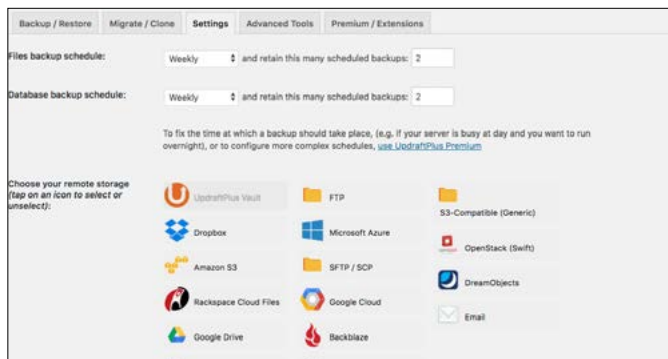
Uno dei motivi della straordinaria popolarità di WordPress è il suo enorme ecosistema di plug-in. Con più di **60.000 add-on** ufficiali su **https://wordpress.org/plug-ins/** e altre migliaia di proposte su siti di terze parti, non manca certo l'imbarazzo della scelta... anche troppo! Orientarsi infatti non è facile e adottare lo strumento sbagliato può far perdere tempo e persino mettere a rischio la sicurezza del sito. Con i **plug-in** giusti, d'altro canto, potete migliorare ogni aspetto del vostro progetto WordPress e introdurvi funzioni che lo faranno spiccare sulla concorrenza. Se vi serve una funzione o la soluzione a un problema e non la trovate in WordPress, inoltre, probabilmente c'è un **add-on** già pronto che fa quello che vi serve in un battibaleno. Quelli presentati in questo articolo si distinguono per la loro efficacia nel facilitare dei compiti utili, che spaziano dall'**email marketing** all'integrazione di **snippet** di codice. Tutti inoltre possono essere utilizzati gratuitamente per le loro funzioni di base. Alcuni sono molto noti, altri meno, ma tutti vi permetteranno di ottenere risultati migliori e di lavorare in modo più efficiente.

Mai più link interrotti

Con questo **plug-in** Open Source potete monitorare e verificare tutti i collegamenti interni ed esterni del vostro sito alla ricerca di quelli non funzionanti. Correggendoli migliorerete la **SEO** e l'esperienza dell'utente. I link si possono modificare direttamente dalla pagina di **Broken Link Checker**, senza aggiornare manualmente ogni occorrenza.

INFORMAZIONI

BROKEN LINK CHECKER

Web: **https://wordpress.org/plug-ins/broken-link-checker/**
Versione: **1.11.18**

Backup per non perdere nulla

UpdraftPlus** consente di fare backup completi, manuali o programmati di tutti i file, **database**, **plug-in** e temi di **WordPress** e di ripristinarli direttamente dal suo pannello di controllo. Offre varie opzioni di archiviazione sul **cloud** tra cui **Dropbox**, **Google Drive** e **Amazon S3** e la versione gratuita è pienamente funzionale.

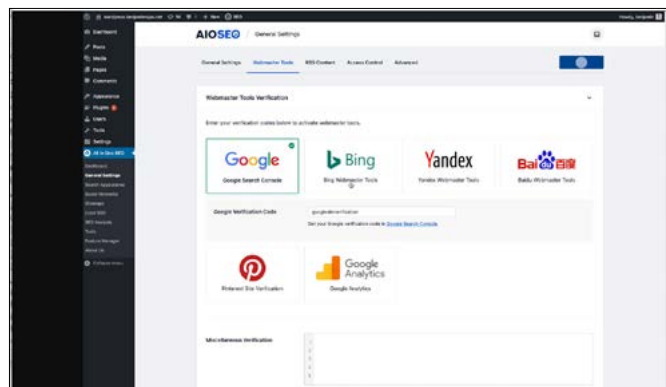
INFORMAZIONI

UPDRAFTPLUS

Web: **https://it.wordpress.org/plug-ins/updraftplus/**
Versione: **1.22.23**

Ottimizzare per i motori di ricerca

La SEO può diventare molto più facile con gli strumenti giusti

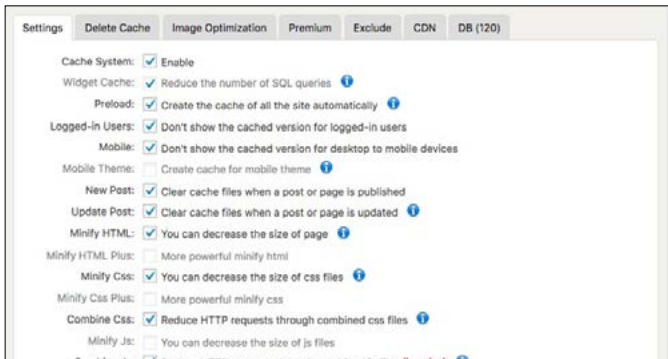
All in One SEO** è un programma Open Source con anche un livello **Pro** a pagamento. Si tratta di un **plug-in** ricco di funzioni e di strumenti di **marketing** che vi permette di implementare strategie di ottimizzazione per i motori di ricerca (in inglese **Search Engine Optimization** o **SEO**) in breve tempo senza ricorrere alla consulenza di specialisti. La SEO è fondamentale per il posizionamento di un sito nella pagina dei risultati dei motori di ricerca e saperla ottimizzare significa avere più visitatori. **AIOSEO** si concentra molto sull'esperienza dell'utente ed è adatto sia ai **marketer** e sviluppatori esperti sia ai neofiti. Offre una procedura guidata di configurazione della SEO in WordPress per aiutarvi a ottimizzare le impostazioni del vostro sito Web in base alle esigenze del vostro settore. Permette di configurare rapidamente aspetti come l'implementazione del file **sitemap.xml**, le parole chiave, l'integrazione con i **social media**, la connessione alla **console per webmaster**, l'applicazione dello **schema Markup** e molto altro. Il plug-in consente di gestire nel dettaglio moltissime impostazioni,

All in One SEO è utilizzato per milioni di siti e offre strumenti molto granulari per gli esperti insieme a soluzioni guidate per i neofiti

il che potrebbe confondere gli utenti meno esperti, ma la presenza di flussi di lavoro creati dal team del progetto e facili da seguire viene in loro soccorso per numerose applicazioni. A differenza di molti altri programmi, All In One SEO include anche nella versione gratuita funzionalità per ottimizzare i siti di **e-commerce**, anche se per avere il supporto avanzato per **WooCommerce** è necessario passare all'edizione **Pro**.

INFORMAZIONI

ALL IN ONE SEO

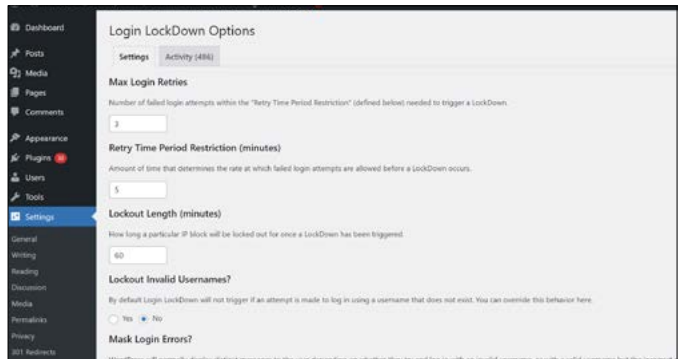
Web: **https://it.wordpress.org/plug-ins/all-in-one-seo-pack/**
Versione: **4.2.5.1**

Caricamenti più veloci

Un pratico **plug-in** che genera **file HTML statici** dal vostro sito dinamico. Il vostro server Web potrà quindi servire quel file invece di elaborare gli **script PHP** di **WordPress**, relativamente più pesanti e impegnativi in termini di utilizzo di **RAM** e **CPU**. Il vostro sito risulterà quindi più veloce da caricare.

INFORMAZIONI

WP FASTEST CACHE

Web: **https://it.wordpress.org/plug-ins/wp-fastest-cache/**
Versione: **1.0.6**

Bloccare attacchi brute force

Per prevenire la rilevazione delle password e gli attacchi di tipo **brute force**, **Login LockDown** registra l'**indirizzo IP** e il **timestamp** di ogni tentativo di accesso fallito. Se viene rilevato più di un certo numero di tentativi in un breve periodo di tempo dallo stesso **range di IP**, la funzione di **login** viene disabilitata per tutte le richieste da quell'indirizzo IP.

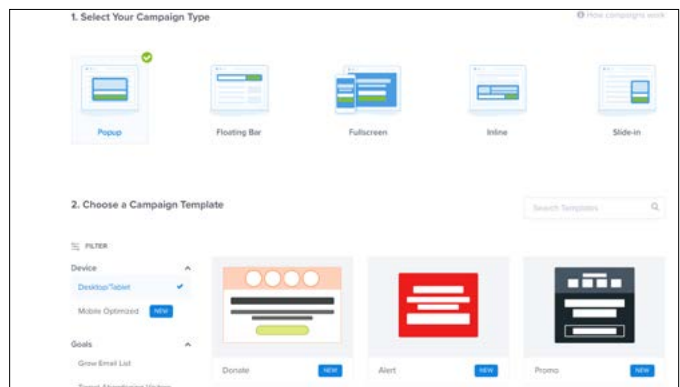
INFORMAZIONI

LOGIN LOCKDOWN

Web: **https://it.wordpress.org/plug-ins/login-lockdown/**
Versione: **1.83**

Non perdere i visitatori del sito

Creare pop-up e moduli per generazione di lead e offerte

Portare gli utenti sul proprio sito richiede molto lavoro, ma mantenere e monetizzare questi contatti non è da meno. **OptinMonster** permette di creare facilmente campagne **pop-up** per le vostre offerte, moduli di iscrizione a **newsletter** via **email** e altri tipi di finestre interattive per il vostro sito. Possono essere utili per promuovere ogni genere di servizio. Portare i vostri utenti a sottoscrivere qualche opzione **opt-in** è un modo per fidelizzarli e rimanere in contatto con loro. Con il termine **opt-in** si definiscono i metodi con cui una persona può esprimere il consenso al ricevimento di informazioni su prodotti o servizi. Potrebbe, per esempio, lasciarvi la sua email per rimanere informata delle novità su un vostro prodotto o per ricevere la vostra **newsletter**. Diventa così quello che nel marketing si chiama un **lead**, ossia un potenziale cliente interessato a quel che offrite. Con OptinMonster potete creare delle campagne personalizzate per generare lead con un editor **drag and drop**, aggiungendo anche elementi dinamici come **timer**

Scegliete il tipo di campagna, il **template** che vi piace di più, il dispositivo su cui verrà visualizzato il sito e molto altro per creare le vostre operazioni di **marketing**

di conto alla rovescia, senza dover scrivere codice. Il programma offre centinaia di modelli per aiutarvi a risparmiare tempo. Il plug-in OptinMonster include un abbonamento gratuito che offre tutte le soluzioni disponibili nell'abbonamento **Basic** con delle limitazioni (fino a **300 impression** al mese e tre campagne). Ci sono poi versioni più potenti (**Pro** o **Growth**).

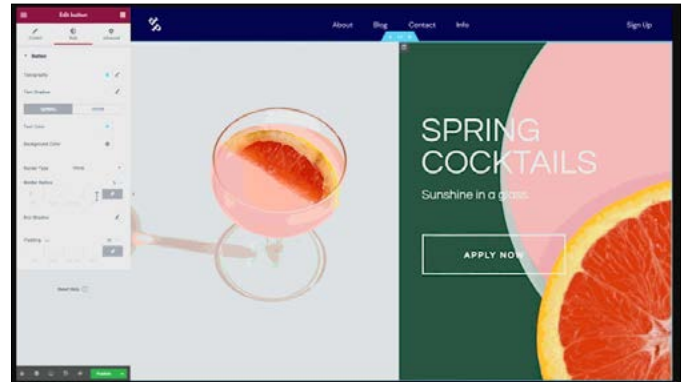
INFORMAZIONI

OPTINMONSTER

Web: **https://it.wordpress.org/plug-ins/optinmonster/**
Versione: **2.10.0**

Siti perfetti con Elementor

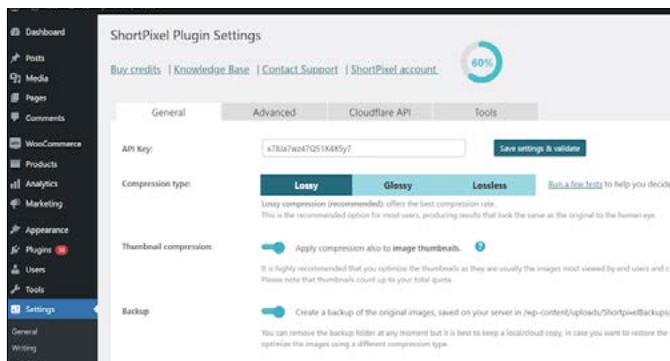
Creare progetti d'impatto in modo intuitivo, anche senza programmare

Con più di 5 milioni di installazioni attive stimate e una valutazione di **4,7** stelle su **5**, **Elementor Website Builder** ha saputo farsi notare. Permette infatti di creare siti Web di livello professionale in modo rapido e intuitivo (senza bisogno di scrivere codice) pur mantenendo un controllo completo su ogni elemento. Il suo **editor drag and drop** per la **modifica in linea** facilita infatti il lavoro a creatori di siti a ogni livello di esperienza. Si possono sfruttare i **Website Kit** completi per creare siti personalizzati velocemente, oppure potete utilizzare la libreria per importare nel vostro sito Web modelli di pagine singole, blocchi o **pop-up**. È anche possibile creare i propri modelli riutilizzabili. Se sapete programmare, poi, potete creare le vostre funzionalità utilizzando l'**API** per gli sviluppatori. Elementor è inoltre Open Source, il che vi dà accesso non solo al codice sorgente ma anche a una **community** molto vasta (**https://elementor.com/community/**) con infinite possibilità di imparare

L'editor di **Elementor** permette di creare le pagine trascinando e modificando tipologie di elementi e ci sono modelli sostanzialmente per qualsiasi cosa

e confrontarsi. Ci sono inoltre decine di **widget** e funzioni pronte all'uso e sul sito dei componenti aggiuntivi di terze parti trovate **900 add-on** che vi permettono di ottenere praticamente qualsiasi funzionalità possa venirvi in mente, anche se di nicchia. Il **plug-in** è gratuito ma ci sono abbonamenti a pagamento con maggiori funzioni di personalizzazione, ideali per i professionisti.

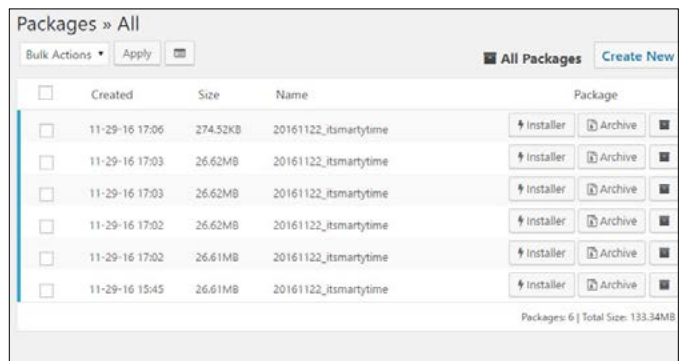
INFORMAZIONI

ELEMENTOR WEBSITE BUILDER
Web: **https://it.wordpress.org/plug-ins/elementor/**
Versione: **3.7.8**

Ottimizzare le immagini

ShortPixel è un **plug-in** facile da usare e leggero per l'ottimizzazione delle immagini. Può comprimere tutte le immagini e i documenti **PDF** presenti nel sito con un solo click e, una volta installato, ridimensiona e ottimizza automaticamente **in background** le nuove immagini. Viene fornito con **100** crediti al mese gratuiti.

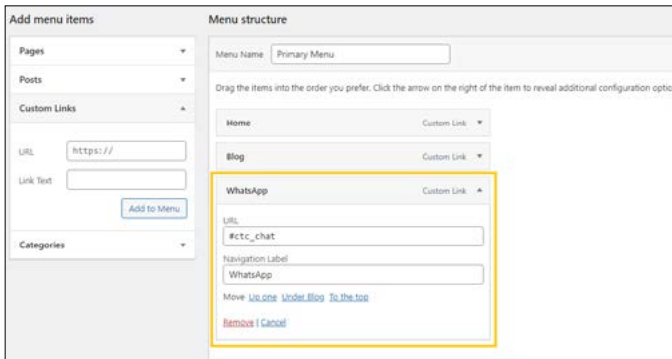
INFORMAZIONI

SHORTPIXEL IMAGE OPTIMIZER
Web: **https://it.wordpress.org/plug-ins/shortpixel-image-optimiser/**
Versione: **5.0.9**

Migrazione senza downtime

Duplicator permette di spostare, **migrare** o clonare un sito **WordPress** tra **domini** o **host** senza interruzioni del servizio (**downtime**). Raggruppa tutti i **plug-in**, i temi, i contenuti, i database e i file di WordPress del sito in un semplice archivio **.zip** chiamato "pacchetto". È Open Source ma esiste una versione **Pro** a pagamento.

INFORMAZIONI

DUPLICATOR
Web: **https://it.wordpress.org/plug-ins/duplicator/**
Versione: **1.5.0**

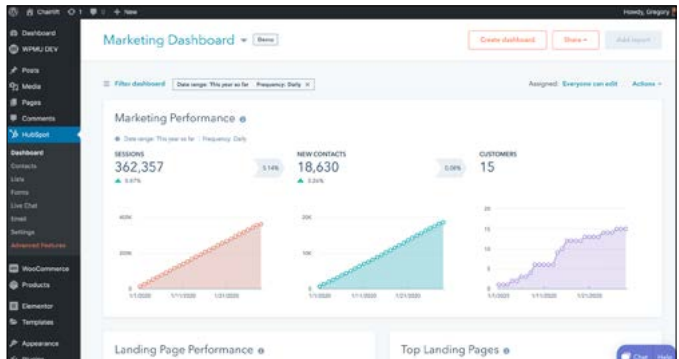
Chiacchierare con i visitatori

Se volete consentire ai visitatori del vostro sito di entrare in contatto diretto con voi, potete integrare questo **plug-in** con cui possono farlo tramite **WhatsApp** o **WhatsApp Business** con un solo click. Personalizzabile per adattarsi al **design** del vostro sito e ricco di funzioni, è Open Source e ben supportato.

INFORMAZIONI

CLICK TO CHAT

Web: **https://it.wordpress.org/plug-ins/click-to-chat-for-whatsapp/**

Versione: **3.12.2**

Tutto il marketing in uno strumento

HubSpot racchiude, in un unico **plug-in**, strumenti avanzati per la gestione delle relazioni con i clienti (**CRM**), l'**email marketing** e l'analisi dei dati. Offre anche la possibilità di integrare una **chat** e dei moduli di interazione con i visitatori del vostro sito. Gratuito e con opzioni avanzate a pagamento, è molto potente ma va studiato un po'.

INFORMAZIONI

HUBSPOT

Web: **https://it.wordpress.org/plugins/leadin/**

Versione: **9.0.123**

WooCommerce per vendere

La soluzione più apprezzata per creare siti di commercio elettronico

Se dovete creare un sito di **e-commerce** e cercate online quale sia la soluzione migliore per farlo, **WooCommerce** risulta sicuramente la scelta più popolare. Gratuito e Open Source (è sviluppato e supportato da **Automattic**, i creatori di **WordPress**), offre tutti gli strumenti per sviluppare siti funzionali e professionali, è flessibile e mette a vostra disposizione una **community** globale. Il fatto che sia Open Source garantisce anche che siate i proprietari di tutto ciò che riguarda il vostro negozio e che non dobbiate mai preoccuparvi che una piattaforma di terze parti chiuda portando con sé preziosi dati. WooCommerce aiuta a curare ogni aspetto di un sito di vendita online sia di beni fisici sia digitali, dalla gestione degli ordini e del carrello, ai rimborsi e alle spedizioni. Il sistema **WooCommerce Payments** (disponibile in vari Paesi tra cui l'**Italia**) permette inoltre di gestire facilmente e in sicurezza diversi tipi di pagamento, sfruttando oltre **100 gateway** tra cui **PayPal**. Per chi non è un tecnico, i vari temi, **template** e strumenti pronti sono la soluzione, mentre per chi programma è disponibile una rapida **interfaccia a riga di comando**. Potete inoltre integrare servizi con **API REST** e **Webhook**, oltre a realizzare blocchi di contenuti personalizzati con **React**. Il progetto ha una documentazione ricca e nel suo **Marketplace** (**https://bit.ly/3yugHL1**) sono disponibili centinaia di estensioni gratuite e a pagamento.

La possibilità di usare numerosi temi pronti e di integrare i prodotti da vendere con un'interfaccia facile da usare rendono **WooCommerce** utilizzabile da tutti

INFORMAZIONI

WOOCOMMERCE

Web: **https://it.wordpress.org/plug-ins/woocommerce/**

Versione: **6.9.4**

Gallerie di foto e video facili

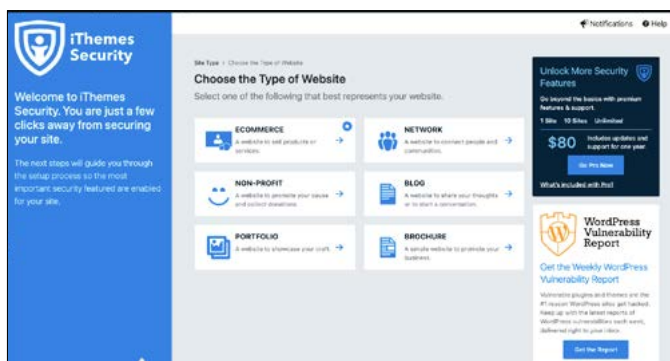
Una soluzione di utilizzo immediato per integrare immagini e video

Con la sua l'interfaccia **drag and drop**, **Envira Photo Gallery** consente di caricare foto e video, riorganizzarli e creare una galleria in pochi minuti. Ottimizza le **query** sul **front-end** e sul **back-end** per dare la massima velocità anche sui dispositivi mobili. Permette di inserire **metadati** e **deeplink** o creare gallerie autonome e offre l'integrazione con **WooCommerce**. È inoltre possibile creare, modificare e sincronizzare le gallerie di immagini e video direttamente all'interno di **Elementor**. Le funzioni di condivisione sui **social network** consentono di **postare** le immagini su piattaforme come **Facebook**, **Twitter** e **Pinterest**, mentre nella vostra galleria video potete aggiungere filmati da **YouTube**, **Vimeo**, **Wistia** e altre fonti oltre, naturalmente, a quelli di cui fate l'**hosting** direttamente. Le opzioni di importazione sono buone e vi permettono di creare gallerie fotografiche dalle vostre raccolte di **Adobe Lightroom**, dal vostro account **Dropbox** e da **NextGen** e si possono caricare anche **archivi .zip**. Altre funzionalità includono la possibilità di aggiungere **tag** alle foto per facilitarne la ricerca, la visualizzazione e il filtraggio. Potete inoltre integrare la protezione con password per impedire l'accesso non autorizzato alle vostre gallerie WordPress. La documentazione del progetto (**https://enviragallery.com/docs/**) è ricca e trovate numerosi altri consigli e guide nel **blog** (**https://bit.ly/3ClCXU0**). Envira Photo Gallery è Open Source e alla versione gratuita si aggiungono quelle a pagamento.

Envira Photo Gallery, con la sua intuitiva interfaccia **drag and drop**, permette di caricare foto e video, organizzarli e creare gallerie in pochissimo tempo

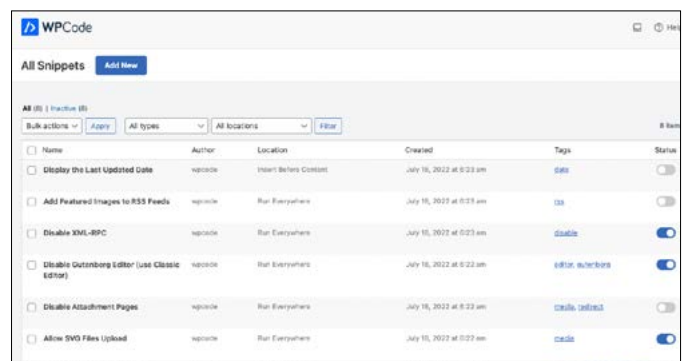
INFORMAZIONI

ENVIRA PHOTO GALLERY
Web: **https://it.wordpress.org/plug-ins/envira-gallery-lite/**
Versione: **1.8.4.7**

Protezione senza stress

Facilitare la protezione del proprio sito anche a chi non è un esperto di sicurezza informatica è lo scopo di **iThemes Security**. Offre profili predefiniti per le principali tipologie di sito. Potete anche integrare funzioni come l'**autenticazione a due fattori** (**2FA**), l'impostazione di requisiti per le password e, con la versione **Pro**, anche **CAPTCHA**.

INFORMAZIONI

ITHEMES SECURITY
Web: **https://it.wordpress.org/plug-ins/better-wp-security/**
Versione: **8.1.2**

Aggiungere codice in WordPress

Con **WPCode** potete aggiungere **snippet di codice** in **WordPress** in modo semplice e senza dover modificare il file **functions.php** del vostro tema. Facilita inoltre l'inserimento di codice come quello di **Google Analytics**, **Pixel** di **Facebook** e altro ancora nell'intestazione, nel piè di pagina e in altre aree del vostro sito WordPress. LXP

INFORMAZIONI

WPCODE
Web: **https://it.wordpress.org/plug-ins/insert-headers-and-footers/**
Versione: **2.0.2**

Tutorial

I nostri esperti offrono i loro consigli di programmazione e di amministrazione del sistema

LA GUIDA DI RIFERIMENTO

Esiste sempre qualcosa di nuovo da imparare in campo informatico, soprattutto in un mondo dinamico come quello di Linux e dell'Open Source. Ogni numero di Linux Pro presenta una sezione dedicata a tutorial realizzati da esperti in moltissimi settori: programmazione, sicurezza, amministrazione di sistema, networking. Troverete informazioni utili sia che siate dei veterani di Linux sia degli utenti alle prime armi. Studieremo con cura anche le applicazioni più diffuse sia in ambito lavorativo che desktop. Il nostro scopo è quello di fornire in ogni numero il giusto mix di argomenti, ma se avete suggerimenti su temi particolari che vorreste vedere trattati, scriveteci via e-mail all'indirizzo tutorial@linuxpro.it

COME RAPPRESENTIAMO LE LINEE DI CODICE

Si presenta spesso la necessità di riportare le linee di codice di un programma. Per favorirne la lettura evidenzieremo le singole linee in questo modo:

```
begin
  mniWordWrap.Checked := not
end
```

Quando una riga di codice supera la lunghezza della colonna la riporteremo su più righe utilizzando la notazione seguente:

```
printf("Vi preghiamo di inserire
una password.");
```

TUTORIAL

ACCADEMIA DEL CODICE

Messaggi segreti a prova di spia

Grazie alla steganografia potrete tutelare al massimo le vostre comunicazioni private e farle passare sotto il naso di chiunque

Si chiama **steganografia** (dal greco per "scrittura nascosta") quella tecnica che permette di occultare un messaggio in forma di file di testo all'interno di un altro file del tutto insospettabile come una innocente immagine **JPG**. Così facendo, solo chi è a conoscenza della presenza del messaggio potrà trovarlo e leggerlo. Ci sono diverse applicazioni e metodi per nascondere un file all'interno di un altro. Uno dei più semplici è quello di comprimere in un file **.zip** il documento da nascondere e di metterlo nella cartella in cui si trova l'immagine all'interno della quale andrà inserito. Dopodiché, aperto il **Terminale** ed entrati nella cartella con i due file, dovrete eseguire una riga di comando con la sintassi:

```
$ cat [nomefile].jpg [nomefile].zip > [nuovofile].jpg
```

Per recuperare il file nascosto, basterà poi eseguire la riga di comando:

```
$ unzip [nuovofile].jpg
```

Se volete anche proteggere il file con una password, allora è necessario usare uno strumento come **Steghide** che si installa da **Terminale** eseguendo semplicemente:

```
$ sudo apt install steghide
```

La sintassi per nascondere un file **TXT** in un **JPG**, ricordando che i due elementi devono trovarsi nella stessa cartella, è la seguente:

```
$ steghide embed -ef [nomefile].txt -cf [nomefile].jpg
```

A questo punto lo strumento vi chiederà di stabilire la password a protezione del documento. Per estrarre il file nascosto dovrete invece eseguire

```
$ steghide extract -sf [nomefile].jpg
```

e poi digitare la password.

ESPLORARE STEG

1 Barra superiore Qui trovate sia i menu che racchiudono tutti i possibili comandi, sia i pulsanti per un uso più rapido.

2 Immagine selezionata Nella sezione a sinistra viene visualizzata l'immagine che avete selezionato per nascondere il file che volete mantenere segreto.

3 Immagine modificata Nella sezione a destra viene invece mostrata l'immagine che già contiene il vostro file nascosto.

4 Stag Options Questa finestra che si apre facendo click sul pulsante **Configuration** vi permette di aggiungere un testo nascosto e di impostare la password.

5 Zoom Utilizzate questi pulsanti per **zoomare** a piacere l'immagine e selezionate **Fit** per proporzionarla in base alle sue dimensioni.

6 Dati Sotto ogni immagine appaiono i suoi dati come la risoluzione in **pixel**.

Strumenti con interfaccia grafica

Fino a questo momento sono stati presi in considerazione solo strumenti che si utilizzano da riga di comando, ma ne esistono anche con una più comoda interfaccia grafica, come **Stegosuite**. Tuttavia, come nei casi visti in precedenza, si tratta di un'applicazione minimalista. Volendo avere a disposizione qualcosa di più complesso, capace di garantire un più alto livello di protezione ai vostri file nascosti, è consigliabile affidarsi a **Steg**. Oltre ad avere un'interfaccia grafica, ha anche il vantaggio di poter creare vari tipi di password con diversi livelli di complessità. Anche se è in inglese, l'interfaccia è piuttosto facile da usare e impiegherete poco tempo a destreggiarvi con essa, riuscendo a nascondere ed estrarre messaggi velocemente. Steg non va installato e, come vedrete nella guida, dovrete rendere eseguibile tramite riga di comando il file **AppImage** che serve ad avviare l'applicazione. Tuttavia, compiuta una volta questa operazione, potrete avviare Steg quando vorrete con un semplice doppio click sul suo file eseguibile. LXP

NASCONDETE UN FILE DI TESTO IN UN'IMMAGINE USANDO STEG

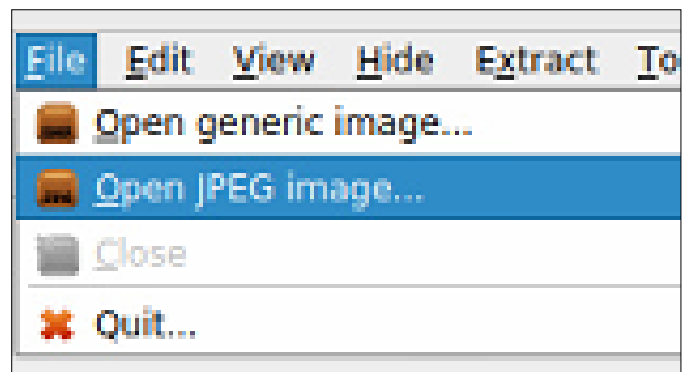
1 Ottenere il file di Steg

Con il vostro programma di navigazione collegatevi all'indirizzo **https://www.fabionet.org/download**. Fate click su **Steg for Linux 64 bit** e poi, nella sezione **Download Link**, su **Steg-1.1.0.0-Linux-x64.tgz**. Quindi aprite la cartella in cui l'avete scaricato.

2 Decompressione del file

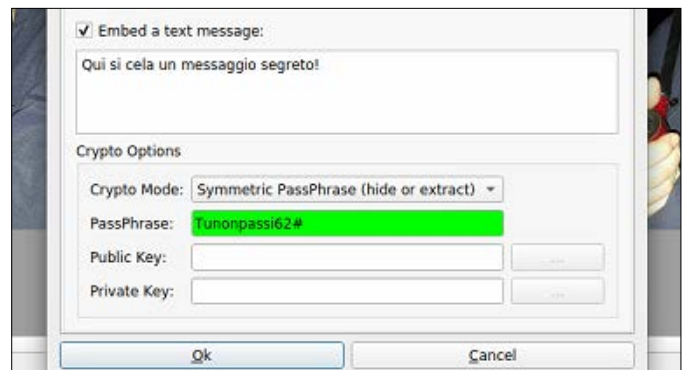
Fate un doppio click sul file appena scaricato e, nella finestra che si apre, premete su **Estrai**. Come cartella di destinazione selezionate **Home** e fate di nuovo click su **Estrai**, quindi chiudete tutte le finestre aperte e avviate una sessione del **Terminale**.

3 Primo avvio di Steg

Eseguite la riga **cd Steg.1.1.0.0-Linux-x64**, poi digitate **chmod a+x Steg*.AppImage** e premete **INVIO** per rendere eseguibile il file di installazione, dopodiché lanciate la riga di comando **./Steg*.AppImage**. Nella finestra **Accept Eula**, scorrete il contratto fino in fondo e fate click su **Yes**.

4 Caricare un'immagine

Fate click su **OK** nella finestra di debug per visualizzare l'interfaccia di **Steg**. Aprite il menu **File** e selezionate **Open generic image** oppure **Open JPEG image**, in base al tipo di immagine che volete usare. Sfogliate le vostre cartelle e, trovata l'immagine, selezionatela e premete su **Open**.

5 Nascondere un file TXT

Aprite il menu **Hide** e fate click su **Hide Data**, poi sfogliate di nuovo le vostre cartelle fino a trovare il file da nascondere. Selezionatelo e premete su **Open** e su **OK** nella finestra di debug. Nella barra superiore fate click sul pulsante **Configuration**, il quarto da sinistra.

6 Creare la password

Nel menu **Crypto Mode** selezionate **Symmetric PassPhrase** e digitate una password con una lettera maiuscola, un numero e un carattere speciale. Potete anche aggiungere un ulteriore messaggio selezionando **Embed a text message**. Fate click su **OK** e salvate il file.

ANONYMOX

Crediti: Hacker Journal

Navigare anonimi su Internet

Installate su Firefox e su Chrome l'add-on anonymoX per navigare con più privacy e anche su tutti i siti bloccati per il nostro Paese

Da tempo il movimento decentralizzato di **hacktivismo Anonymous** porta avanti una **cyberguerra** dichiarata contro **Putin** e propone una "cassetta degli attrezzi anticensura". In pratica, si tratta di una serie di strumenti adatti a tutelare la privacy aggirando le limitazioni imposte dal governo sovietico. Certo, molti di questi sono abbastanza noti, ma ce ne sono diversi che, sicuramente, meritano maggiore fortuna e di essere conosciuti dai più. Uno dei tool più interessanti tra quelli consigliati da Anonymous è senza dubbio **anonymoX**. Si tratta di un semplice **add-on** per browser (funziona sia con **Firefox** sia con **Chrome**) che consente di celare il proprio **indirizzo IP** e contemporaneamente la zona da cui ci si collega. È facile sin da subito intuirne le potenzialità: capirete bene infatti che, essendo appunto capace di nascondere la vostra posizione, può essere tranquillamente utilizzato per evitare che venga individuata la propria posizione geografica. Questo plug-in, però, non è utile solo per raggiungere siti bloccati da chi si connette da determinate location, ma consente anche di tutelare fortemente la privacy in quanto evita quello che gli esperti di marketing etichettano come "profilazione avanzata in base alla posizione". In parole povere, la comparsa di annunci o inserzioni mostrate a video grazie alla registrazione dell'IP e dei **cookie**. Provate, per esempio, a cercare su Google un ristorante o un bar senza inserire alcuna località. Il motore di ricerca vi restituirà i risultati mostrandovi le attività nelle vostre vicinanze. Beh, provate poi a fare la stessa operazione dopo aver installato anonymoX!

FACILE DA USARE

"Permette di raggiungere quasi gli stessi risultati ottenibili usando una VPN con un add-on nel browser"

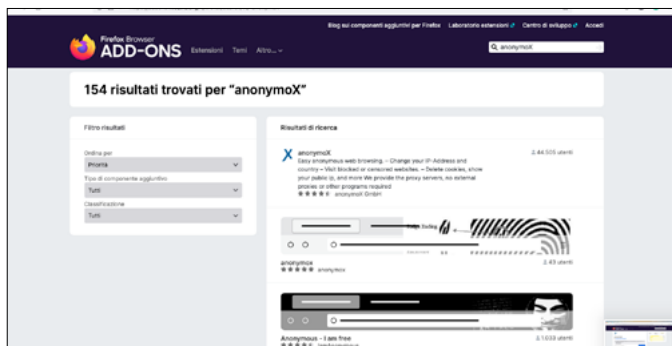
Indirizzo IP: 62.10.102.131
Codice Nazione: Italy
Regione: Italy
Latitudine: Italy
Longitudine: Italy

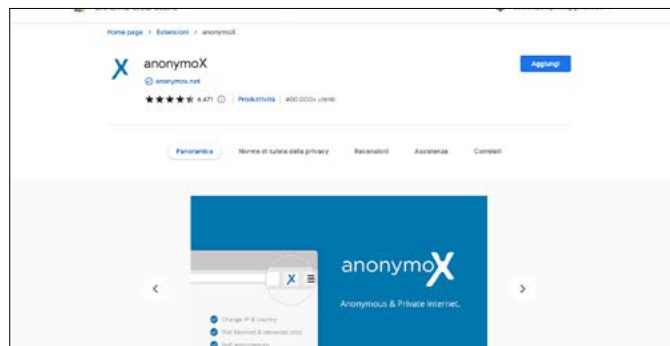
Indirizzo IP: 159.253.145.183
Codice Nazione: Netherlands
Regione: Netherlands
Latitudine: Netherlands
Longitudine: Netherlands

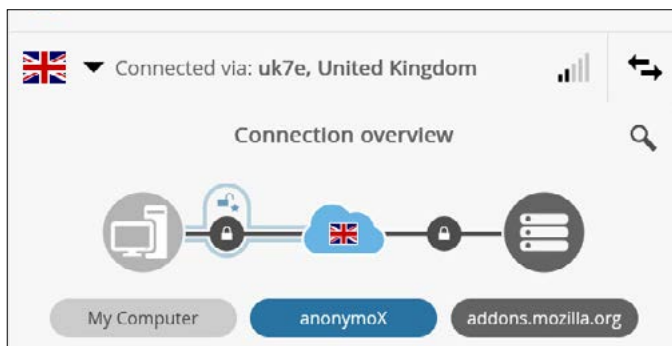
Sulla sinistra, l'accesso a **https://bit.ly/ilmioip** con l'**add-on** disabilitato. Sulla destra, lo stesso sito ma **anonymoX** attivo. Potete vedere l'**IP** e la zona di connessione cambiate

Sviluppi interessanti

Come già detto, l'utilizzo di un software come questo offre grandi potenzialità: non solo consente di visitare siti bloccati a visitatori esteri o interni di un Paese, ma permette anche di raggiungere, praticamente, gli stessi risultati ottenibili usando una VPN solo installando banalmente un add-on nel browser. Senza contare che offre l'opportunità di sfruttare servizi disponibili in altri stati e non presenti da noi, il che può risultare sicuramente utile. LXP

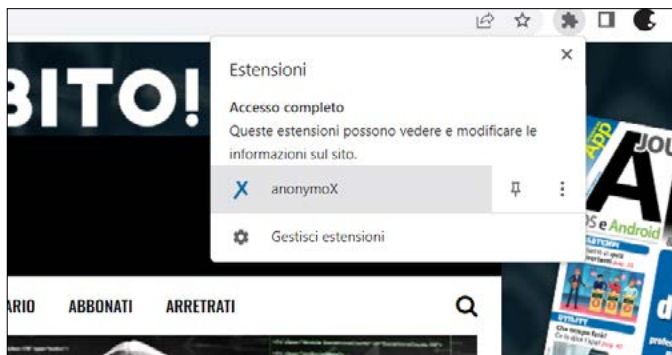
ACCEDERE AI SITI DA UN PAESE DIVERSO CON ANONYMOX

1 Installare l'add-on in Firefox

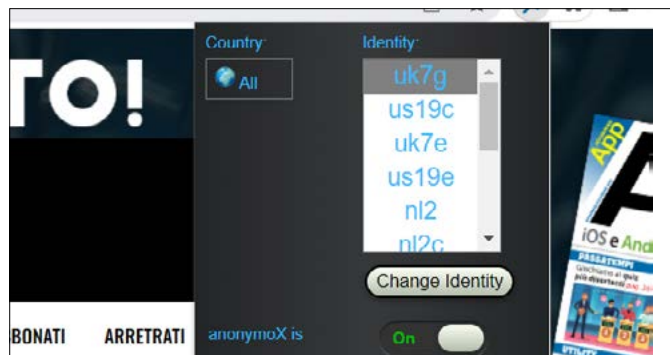
Se navigate con **Firefox**, le prime operazioni da fare sono collegarsi a **https://addons.mozilla.org** e cercare il nome dell'**add-on**, ovvero **anonymoX**. Poi, date un **Invio** e fate click sul primo risultato, l'add-on contrassegnato con una **X** azzurra. Infine, scegliete il pulsante **Aggiungi a Firefox** e, subito dopo, **Installa**.

2 Aggiungere anonymoX in Chrome

Se utilizzate invece **Chrome**, dopo aver fatto l'accesso al vostro account **Google**, collegatevi a **https://chrome.google.com/webstore/category/extensions** e cercate nello **store** lo stesso nome. A differenza che in **Firefox**, troverete un solo risultato per **anonymoX.** Entrate nell'estensione e fate click su **Aggiungi**.

3 Anonimi con Mozilla

Sul browser di **Mozilla**, **anonymoX** appare con la sua **X** azzurra sulla barra dei menu. Facendoci click sopra si apre una finestra che mostra il collegamento del computer con il sito in cui si sta navigando. Spostate la levetta in alto su **Active**. Questa diverrà azzurra e il disegno del collegamento cambierà.

4 Scegliere la nuova località

Facendo click sulla freccia e aprendo il menu a comparsa potete modificare la località dalla quale volete "collegarvi" al sito che volete visitare in completo anonimato. Come vedete, sono disponibili ben sette **location**: tre inglesi e quattro olandesi. Per aggiungerne altre bisogna passare alla versione **Premium** dell'**add-on**.

5 In incognito su Google

Per aggiungere l'estensione a **Chrome** dovete accedere a Google e collegarvi al **Chrome Web Store** digitando **https://chrome.google.com/webstore/**. Nel box di ricerca digitate il nome dello strumento e, dopo averlo individuato, fate click su **Aggiungi** e poi su **Aggiungi estensione**.

6 Cambiare identità a piacimento

Così come con **Firefox**, anche con il browser di **Google** è possibile modificare la località di connessione. Quelle disponibili (dodici per la versione **free**) appartengono a tre Paesi: **USA**, **Olanda** e **Gran Bretagna**. Per passare da una all'altra vi basterà premere il pulsante **Change Identity** presente sotto l'elenco.

TERMINALE

Crediti: alexandruburlacu.github.io

Logging, tracing e monitoraggio

I log sono molto importanti per tener traccia di cosa succede nel computer e poter fare debugging. Ma non sono l'unico strumento a vostra disposizione

Per qualsiasi sistema utilizzato in produzione, i **log** sono essenziali. Sono una registrazione degli eventi del sistema, come delle voci in un diario su quello che è accaduto. Indicano il momento in cui un evento si è verificato, una descrizione e magari anche un contesto. Vi aiutano a scoprire cosa stesse succedendo un attimo prima dell'arresto di un'applicazione o di un'attività malevola. Ma c'è un'arte nel creare dei buoni log.

I criteri di un buon messaggio di log

Ci sono alcuni criteri per progettare log davvero utili. Innanzitutto devono essere gerarchici: dobbiamo rispettare la distinzione tra **DEBUG**, **INFO**, **WARNING**, **ERROR** ed eventualmente altri livelli. Non bisogna affollare il sistema con allarmanti log di WARNING quando sarebbero più appropriati dei log di INFO o DEBUG. Non vanno nemmeno sovraccaricati i log con troppe informazioni. Detto questo, è bene che un log ERROR registri il maggior numero possibile di informazioni per facilitare il **debugging**. Utilizzate i log a livello DEBUG per registrare informazioni sulle impostazioni utilizzate dal programma, o anche sul tempo o le risorse impiegate da una **subroutine**, ma non abusatene. I log INFO dovrebbero contenere dati come, per esempio, le informazioni su una chiamata a un **route handler** di primo livello in un **server HTTP**. Un altro principio dei log è che devono essere informativi. Devono registrare tutto ciò che può aiutare nel debugging del sistema. Se si verifica un errore, è opportuno memorizzare il **traceback**. Inoltre, sarà utile registrare il contesto in cui è comparso il problema, ossia le variabili circostanti che potrebbero esservi collegate. Se il sistema funziona con più processi o è **multithread**, è importante registrare i **PID** e gli **ID** dei **thread**. Un'altra caratteristica di un buon log è quella di essere filtrabile: i log sono fatti per essere analizzati. Rendeteli il più possibile interrogabili. Considerate la possibilità di formattarli come documenti **JSON** e non abusate della **nidificazione**. Se il JSON è troppo **annidato**, infatti, diventa difficile analizzarlo, vanificando il suo scopo. Per esempio, **Elasticsearch** non è in grado di indicizzare correttamente file JSON con due o più livelli di annidamento. In altre parole, qualcosa di simile all'esempio che segue può essere indicizzato:

```
{"timestamp": "2021-05-18T21:09:54Z", "level":
"error", "msg": "si è verificato un problema"}
```

Anche questo esempio è indicizzabile:

```
{"timestamp": {"date": "10 Nov, 2022", "time":
"11:30:30am"}, "level": "error", "msg": "si è verificato
un problema"}
```

Se però avete qualcosa di simile a:

```
{"timestamp": {
   "date": "10 Nov, 2022",
   "time": [11, 30, 30, 124]
```

Progettare correttamente la struttura di **logging** è essenziale per tenere traccia di ciò che avviene in un ecosistema

TIP

Vi siete mai chiesti perché i log si chiamino così? Il termine in inglese indica un tronco di legno. I marinai del XVIII secolo tiravano in mare dei "log" con fissata una fune con nodi a distanza regolare e, contandoli, capivano la velocità della nave. Il logbook diventò poi il registro di navigazione delle navi, con velocità, altri dati e gli eventi del viaggio. Per questo la parola log è arrivata a indicare il "diario" degli eventi avvenuti in un computer.

```
    },
  "level": "error",
  "msg": "si è verificato un problema",
  "context": {
    "some_key_for_multiple_values": []
    }
}
```

Elastic tratterà gli elementi profondamente annidati come **stringhe** e non sarà facile filtrare e aggregare questi log. Un altro buon formato è l'**NCSA Common log format** ma, se possibile, scegliete JSON, perché la maggior parte degli strumenti di analisi dei log lo utilizza. Ecco degli esempi di buoni e cattivi log.

```
Cattivo log (1): [2021-05-17 12:30:30] ERROR: KeyError
// La versione JSON sarebbe altrettanto inutile
Cattivo log: {"datetime": {"date": "10 Nov, 2022",
"time": "11:30:30am"}, "type": "ERROR", "msg": "Si è
 verificato un errore KeyError nella funzione una_
 funzione"}
Log migliore: {"timestamp": "2021-05-18T21:09:54Z",
"level": "error", "pid": 1201, "traceback": <your
traceback as a string>, "msg": "KeyError: 'key
name'"}
```

Accedere ai dati e analizzarli

Una volta che si dispone di log ben scritti, si deve decidere come accedervi e analizzarli. Queste scelte devono essere guidate anche dalla fase e dalla scala del sistema. In altre parole, se avete un'applicazione che serve poche centinaia di persone, è meglio non creare un'infrastruttura complessa. Le fasi sono indicativamente tre: la raccolta/l'invio dei log, la loro memorizzazione e la loro elaborazione/analisi. Partiamo dalla prima. Bisogna salvare i log da qualche parte (e non limitarsi a stamparli su **stderr/stdout**) quindi è necessario pensare a dove scriverli. Potrebbe essere un file, o un **Syslog**, per esempio, o si potrebbero anche scrivere in un **socket TCP** o **UDP**, inviandoli a un server di log. In realtà, tutte queste scelte sono piuttosto valide. Purché non si blocchi il **thread** in cui avviene l'azione, non ci dovrebbero essere problemi; in caso contrario, ci si deve preparare a un calo delle prestazioni. Per quanto riguarda l'archiviazione per un'applicazione semplice, lasciarli come file può andare bene per un po', ma alla fine è consigliabile una soluzione con supporto per l'**indicizzazione**. Una volta che si dispone di più servizi, si può pensare a un server di log centralizzato, come un **cluster ELK** (**Elasticsearch**, **Logstash**, **Kibana**), con una o più istanze di **Elastic**. Dovreste anche impostare la **rotazione** dei file di log per evitare di ritrovarvi con un singolo file di testo da **10 GB**. A un certo punto, dovrete anche pensare alla compressione dei log e, probabilmente, al **log shipping**, il che significa trasferire i file di registro dal luogo in cui sono stati creati a quello in cui verranno analizzati e conservati sul lungo termine. Per la spedizione dei file di log è consigliabile utilizzare **TCP** o **HTTP** rispetto a **UDP** e ad altri protocolli. Il motivo è che con UDP si potrebbero perdere i log durante il trasferimento, perché non c'è modo di ritrasmettere i pacchetti persi e non c'è controllo di flusso, il che potrebbe causare la perdita dei pacchetti. Inoltre, le dimensioni dei messaggi sono limitate a **65 KB** di dati, o anche meno, a seconda delle impostazioni di rete, il che spesso non è sufficiente. Inoltre, i **firewall** aziendali a volte bloccano questo tipo di traffico.

L'importanza delle notifiche

Utilizzare un sistema che vi avvisi dei problemi quando insorgono è sicuramente utile ma vanno considerati alcuni fattori. Innanzitutto, se ne avete la possibilità, impostate delle soglie per gli avvisi, in modo da non ricevere una notifica ogni volta che si verifica un problema, anche minimo. Magari si tratta di un evento unico (non critico)

» SCEGLIERE IL FORMATO DEI LOG

Come si è visto nel corpo dell'articolo, due ottimi formati per i **log** sono **JSON** e **NCSA Common log format**. JSON (**JavaScript Object Notation**) è un formato per lo scambio di dati facile da leggere e scrivere per gli esseri umani e al contempo semplice da generare e analizzare per le macchine. Si basa su un sottoinsieme del linguaggio di programmazione **JavaScript**, **Standard ECMA-262 Terza Edizione**. Anche se JSON è completamente indipendente dal linguaggio di programmazione, sfrutta delle convenzioni conosciute da chi scrive codice in linguaggi della famiglia del **C**, come **C**, **C++**, **C#**, **Java**, **JavaScript**, **Perl**, **Python** e molti altri, il che lo rende di immediato utilizzo per molti. L'NCSA Common log è un formato di file di testo standardizzato utilizzato dai server Web per la generazione di file di log. Può essere facilmente analizzato da diversi programmi di analisi Web, come per esempio **Webalizer** e **Analog**. Ogni riga di un file memorizzato in questo formato ha la seguente sintassi:

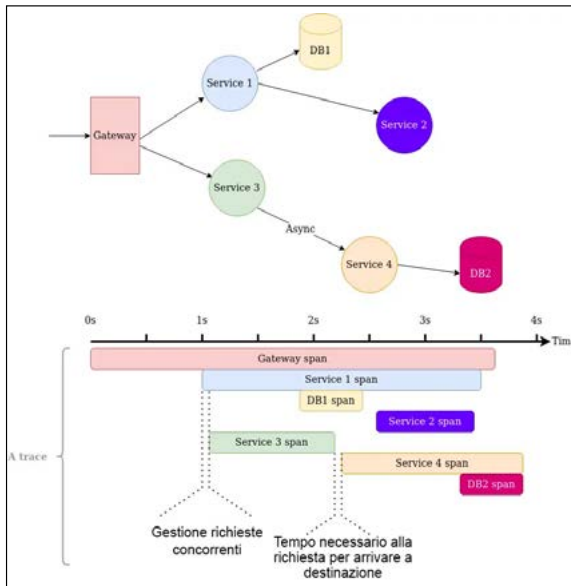
```
host ident authuser date request status
bytes
```

In genere l'NCSA Common log è considerato migliore per i sistemi più piccoli, in cui è possibile fare ricerche nei log con **grep** e altri strumenti. Qualunque sia il formato che sceglierete, è fondamentale essere coerenti per l'intero sistema.

```
{
    "name": "Tina",
    "surname": "Rossi",
    "active": true,
    "favoriteNumber": 12,
    "birthday": {
        "day": 25,
        "month": 4,
        "year": 1975
    },
    "languages": [ "it", "en" ]
}
```

La descrizione di una persona in formato **JSON**

TIP

L'antifragilità è una proprietà dei sistemi che aumenta la capacità di prosperare a fronte di sollecitazioni, fattori di stress, volatilità, disordine, errori, guasti o attacchi. Il concetto è stato sviluppato da Nassim Nicholas Taleb nel suo libro Antifragile ed è stato applicato a vari campi della scienza oltre che all'informatica.

In questo diagramma potete vedere il funzionamento del **tracing** che mette in correlazione i dati raccolti sui diversi servizi

e non c'è bisogno di preoccuparsi, mentre se la problematica si presenta frequentemente è meglio essere avvisati. Un altro aspetto da tenere in considerazione, quando si parla di avvisi, è la possibilità di prevedere invii diversi in successione. Innanzitutto, mandate un avviso via email. Se non viene intrapresa alcuna azione, inviatelo a un gruppo di **chat** del team responsabile. Ancora nessuna attività? Provate con un messaggio diretto a un ingegnere o a un responsabile tecnico. Infine, cercate di aggregare il materiale: non c'è bisogno di inoltrare decine di email o messaggi su **Slack** sullo stesso problema. Per quanto riguarda gli strumenti da utilizzare per le notifiche, una buona opzione è **Sentry** (che offre anche un livello gratuito) ed è possibile impostare degli avvisi anche in **Kibana**.

Usare i file di log con criterio

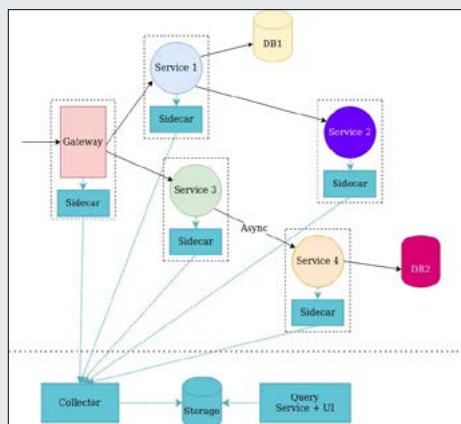
Quando si entra nel mondo della **telemetria** e del monitoraggio delle prestazioni, inizialmente si ha la tentazione di usare i log per tutto. Per esempio, se il sistema è lento, si può **loggare** il tempo di esecuzione, il numero di richieste e così via. In linea di principio si potrebbe fare, ma è meglio avere un'infrastruttura diversa, per non complicare le cose. Con "complicare le cose" intendiamo che, per esempio, potreste voler impostare il monitoraggio delle prestazioni non solo a livello di **route controller**, per vedere quanto tempo impiegano le richieste a essere gestite e a ricevere una risposta (assumendo un ipotetico server). Vorrete anche tenere traccia di quanto tempo impiegano le **query al database** per essere eseguite o magari anche le funzioni. A questo punto avrete moltissime informazioni dettagliate ma che sicuramente andranno a sovraccaricare l'infrastruttura di **logging**. Inoltre, anche se tutto funzionasse senza problemi, i vostri **pattern** di lettura e scrittura sarebbero diversi. Le query di analisi dei log possono essere molto più complesse di quelle richieste per il monitoraggio delle prestazioni. Inoltre quest'ultimo ha solitamente messaggi più piccoli che devono essere registrati con una **latenza** inferiore. In definitiva, è meglio creare un'infrastruttura dedicata a questo scopo. La cosa più semplice è ovviamente usare il logging a livello di tracciamento e avere un'infrastruttura dedicata al monitoraggio delle prestazioni. Questo approccio, però, funziona solo su piccola scala, dove in realtà non è nemmeno necessario il monitoraggio delle prestazioni. Quando il sistema cresce, si potrebbe optare per un tipo di log più ristretto, magari qualche **protocollo binario**, dato che si inviano piccoli pacchetti di informazioni molto frequentemente. Poiché, come abbiamo detto, il monitoraggio delle prestazioni ha un modello di scrittura e di interrogazione un po' differente da quello dell'analisi dei log, è consigliabile salvarli diversamente. Le query sono più semplici e mostrano principalmente tendenze, **serie temporali**, valori correnti o alcuni semplici **valori aggregati**, come conteggi, medie, **mediane** e **percentili**. Le scritture sono molto frequenti, ma con pochi dati e poche **metriche** rispetto ai log di traceback, contesti e simili. Questo è il motivo per cui, per esempio, lo stack ELK è più comune nelle infrastrutture di logging, mentre Elasticsearch è in grado di indicizzare e analizzare anche dati molto destrutturati e strumenti come **Grafana** e **Prometheus** sono più comunemente utilizzati per il monitoraggio delle prestazioni. Prometheus, tra le altre cose, contiene un database di serie temporali che è proprio quello che serve per memorizzare e interrogare rapidamente le metriche delle prestazioni. Inoltre, quando si tratta di analisi delle prestazioni, occorre monitorare l'utilizzo del sistema, non solo gli aspetti intrinseci al codice. Se si utilizza Prometheus, è facile farlo.

Un passo oltre il logging e il monitoraggio delle prestazioni

Innanzitutto è importante inquadrare la rete e i sistemi dinamici: anche se intuitivamente non sembra così, una rete di computer è una risorsa condivisa con una capacità limitata. Ciò significa che, se un servizio è molto attivo, influenzerà il **throughput** e la latenza di tutti gli altri. Inoltre, dato che le reti non sono a priori affidabili al 100% e che utilizziamo prevalentemente traffico basato su **TCP**, nella rete ci saranno molti pacchetti (blocchi di dati, ritrasmissioni, pacchetti di protocolli amministrativi). Questo però è solo metà del problema. I nostri servizi dipendono l'uno dall'altro e da terze parti. Quindi se uno di essi è lento potrebbe influenzare gli altri, anche quelli che non interagiscono direttamente con esso. Una metafora che aiuta

a visualizzare questa situazione è una ragnatela. Quando la si tocca da un lato, si increspa dall'altro. È un po' come l'**effetto farfalla** in matematica. E non si tratta di un semplice paragone: è possibile che si verifichi un guasto dovuto al rallentamento di qualche altro servizio. Quindi, come si può monitorare questo aspetto? Si potrebbe pensare di usare i log o i metodi di monitoraggio delle prestazioni visti prima. È un inizio, ma i soli log non bastano, perché non vediamo il quadro completo, in particolare non osserviamo l'interazione tra i servizi ma solo le prestazioni di ciascuno di essi. Abbiamo bisogno di qualcosa di più: il **tracing**. Per avere un buon modello mentale del tracing, si può pensare che sia come il logging ma con un **identificatore di correlazione** che rende possibile combinare i log in una "traccia". Questa può mostrarci, per esempio, come una singola richiesta si estende su più servizi, quanto tempo impiega ogni passo e quanto ne è stato necessario per la comunicazione. Tutto questo può aiutare a scoprire **bug** e **colli di bottiglia** delle **performance** che un semplice strumento di monitoraggio delle prestazioni, o i soli log, non sono in grado di individuare. Il tracing vi aiuterà a trovare i servizi che presentano colli di bottiglia e a volte è utile anche nel debugging dei **sistemi distribuiti**. Va infatti considerato un'estensione degli strumenti di monitoraggio delle prestazioni, piuttosto che dei log.

» APPROFONDIRE IL TRACING

Un'architettura comune per i **sottosistemi di tracing** è una combinazione di componenti **sidecar** (vedi il **Tip** in questa pagina) **collector**, **storage** e **presenter**, oltre alla **libreria client**. Se si vuole usare il tracing in una configurazione **serverless**, la situazione si complica; una soluzione potrebbe essere quella di **bypassare** il **sidecar** e inviare i dati direttamente al **collettore**, ma così si perdono alcune caratteristiche interessanti. Il tracing, in generale, è un argomento molto vasto. Potete trovare una trattazione dettagliata in un'analisi (in inglese) dell'**università Carnegie Mellon** all'indirizzo **https://bit.ly/3ClQkrP** e in questo articolo (**https://ubr.to/3rGsjqv**) sul blog di **Uber**. Nel primo sono discussi vari argomenti importanti come le strategie di campionamento delle tracce e la loro visualizzazione, mentre il secondo illustra un caso di applicazione pratica del tracing, ossia il suo utilizzo nei sistemi di Uber.

Un esempio di utilizzo dei componenti che possono essere coinvolti in un **sottosistema di tracing**

Come sfruttare al meglio il tracing

Lo scopo principale del tracing è quello di scoprire i problemi di prestazioni e, a volte, di individuare il motivo per cui un'operazione specifica non è andata a buon fine. Le tracce si possono usare come log ma non bisogna sovraccaricarle di informazioni, altrimenti l'infrastruttura di raccolta, archiviazione e analisi ne risentirà. È anche importante capire come strutturare il tracing, La soluzione più semplice da attuare è utilizzare strumenti che **patchano** automaticamente le vostre dipendenze, come i **client di database**, i **server Web** e i **client HTTP/RPC**. Offrono in genere impostazioni predefinite ragionevoli. Se si desidera avere più controllo, bisogna prepararsi a scrivere un po' di codice, soprattutto se si vuole identificare manualmente cosa viene propagato tra i servizi. Quando dovete aggiungere informazioni ai vostri **span** (le parti che combinate formano una traccia) non inserite l'intero contesto dell'applicazione, ma solo gli elementi più importanti come, per esempio, le configurazioni attuali del vostro sistema. A volte è importante correlare le tracce con i log e per questo si può usare un altro identificatore di correlazione per un'analisi più approfondita del sistema, combinando il tracing con i singoli log. Esistono alcuni strumenti Open Source con un ottimo supporto, come **Jaeger** e **Zipkin**, ma ci sono anche iniziative come **OpenTracing**, **OpenCensus** e la loro "combinazione" **OpenTelemetry**. Esistono inoltre alcuni formati di tracciamento, come **W3C Trace Context** e **Zipkin B3**.

Il valore dell'osservabilità

Nella **teoria del controllo**, la proprietà dell'**osservabilità** di un sistema dinamico determina la possibilità di risalire al suo stato a partire dalla conoscenza del suo **output**. Si sviluppa su uno **spettro** e, a seconda della posizione del vostro sistema, potete utilizzare il monitoraggio e gli avvisi in modo più o meno efficiente. Dobbiamo progettare i nostri sistemi tenendo conto dell'osservabilità e, con tutti i metodi sopra descritti, dovrebbe essere un compito fattibile. Si può pensare all'osservabilità, insieme a una corretta procedura di risposta agli incidenti, come a un modo per rendere il sistema **antifragile** (vedi il **Tip** nella pagina precedente), perché a ogni problema che si verifica, "impara", a livello organizzativo, a essere migliore. In ogni caso, le procedure citate dovrebbero aiutarvi a risolvere anche i bug più oscuri. Naturalmente applicarle non è semplice e richiede di organizzare un'infrastruttura. Se però questo aiuta a ridurre i tempi di risoluzione di un problema da una settimana (o più) a uno, forse due, giorni, probabilmente ne vale la pena. LXP

TIP

I file sidecar, noti anche come file buddy o file collegati, memorizzano dati (spesso metadati) non supportati dal formato di un file sorgente. Ci possono essere uno o più file sidecar per ogni sorgente.

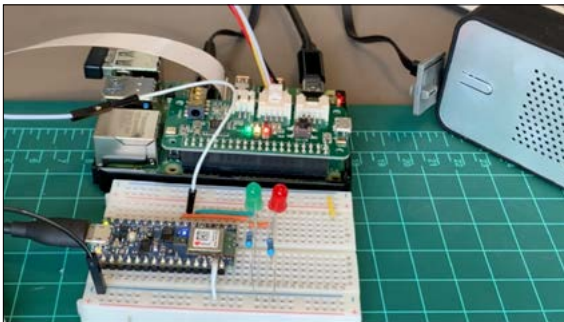
Crediti: MJRoBot (Marcelo Rovai)

Creare un assistente vocale intelligente

Realizzate la vostra versione personale dell'Assistente Google con una Raspberry Pi e un Arduino Nano 33 BLE

Siamo ormai tutti abituati agli assistenti vocali ma vi siete mai chiesti come funzionano? Cercheremo in queste pagine di rispondere a questa domanda emulando l'**Assistente Google** tramite una **Raspberry Pi** e un **Arduino Nano**.
Per cominciare, è essenziale rendersi conto che gli assistenti vocali presenti sul mercato, come **Google Home** o **Amazon Echo**, reagiscono agli esseri umani solo quando vengono "svegliati" da particolari parole chiave (dette anche **wake word**) come **Hey Google** sul primo e **Alexa** sul secondo. In altre parole, l'intero processo di riconoscimento dei comandi vocali si basa su un modello in più stadi. In primo luogo, un microprocessore all'interno del dispositivo Echo o Google Home ascolta continuamente il suono circostante, in attesa che venga individuata la parola chiave. Per questo rilevamento viene utilizzato un modello **TinyML**. Solo quando il dispositivo viene attivato (**fase 2**), i dati vengono inviati al **cloud** e quindi elaborati su un modello più grande.
Per la **fase 1** (**KWS** o **Keyword Spotting**) di questo progetto utilizzeremo un **microcontrollore Arduino Nano 33 BLE Sense** che ha, tra i vari sensori incorporati, un microfono digitale che verrà utilizzato per individuare la parola chiave. Nella **fase 2** si usa una **Raspberry Pi** per contattare i servizi **Google** sul cloud ed eseguire un'attività più complessa dopo l'attivazione da parte di Arduino. Il progetto è suddiviso in due parti: emulazione dell'Assistente Google su una Pi e implementazione di un KWS su Arduino Nano.

COSA SERVE
- **Raspberry Pi 3 Model B**
- **Arduino Nano 33 BLE Sense**
- **ReSpeaker 2-Mics Pi HAT di Seeed Studio**
- **Altoparlante, 2W**
- **Arduino IDE**
- **Google Assistant SDK**
- **Codice: https://www.hackster.io/mjrobot/building-an-intelligent-voice-assistant-from-scratch-2199c3**

Un microprocessore nel dispositivo **Google Home** ascolta sempre, in attesa che la parola chiave venga individuata con un modello **TinyML**

Il progetto emula l'**Assistente Google** su una **Pi** e usa un **Arduino Nano 33 BLE Sense** per riconoscere la parola chiave

Emulare l'Assistente Google su una Pi

Oltre al software che verrà sviluppato per consentire alla Pi di emulare l'Assistente Google, saranno necessari anche alcuni componenti hardware aggiuntivi. Potete installare un microfono e un altoparlante esterni o utilizzare un **HAT** per semplicità. In questo progetto utilizzeremo **ReSpeaker 2-Mics Pi HAT**. La sua installazione è molto semplice: basta collegarlo alla Pi e configurare il **driver** su quest'ultima.

```
sudo apt-get update
sudo apt-get upgrade
git clone https://github.com/respeaker/
  seeed-voicecard.git
```

```
cd seeed-voicecard
sudo ./install.sh
reboot
```

Controllate le schede audio installate sulla Pi. Nel nostro caso la **scheda 0** è l'**HDMI** della Pi, la **1** è il suo **Jack audio** per le cuffie e la **2** è il **ReSpeaker 2-Mics Pi HAT**. La **scheda 2** dovrebbe essere quella predefinita e potete verificarlo in **Preferences > Audio Device Settings**. Questa configurazione si effettua modificando il **file.asoundrc** nella cartella **/home/pi**. Per farlo, eseguite il seguente comando da **terminale**

```
sudo nano /home/pi/.asoundrc
```

e cambiate l'impostazione di **pcm.output** dalla **scheda 2** alla **1**. A questo punto è possibile effettuare alcuni test. Per provare l'uscita audio scrivete:

```
speaker-test -t wav
```

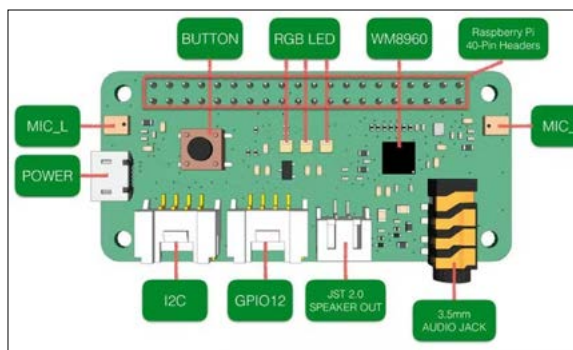
Si dovrebbero sentire "**Front**" e "**Left**" ripetuti sull'altoparlante. Premete **Ctrl + C** per uscire.

Test dell'ingresso audio

Per prima cosa, installate la **libreria Python PyAudio** utilizzata per riprodurre e registrare l'audio sulla Pi:

```
sudo pip install pyaudio
```

Nella pagina Web di PyAudio (**http://people.csail.mit.edu/hubert/pyaudio/**) trovate informazioni ed esempi su come lavorare con la libreria. Utilizzando lo **script** che segue, registrate cinque secondi di audio:

```
import pyaudio
import wave
RESPEAKER_INDEX = 2 #fa riferimento all'ID
del dispositivo di ingresso (scheda 2)
RESPEAKER_RATE = 16000
RESPEAKER_CHANNELS = 2
RESPEAKER_WIDTH = 2
CHUNK = 1024
RECORD_SECONDS = 5
WAVE_OUTPUT_FILENAME = "record_test.wav"
p = pyaudio.PyAudio()
stream = p.open(
   rate=RESPEAKER_RATE,
   format=p.get_format_from_width(RESPEAKER_
WIDTH),
   channels=RESPEAKER_CHANNELS,
   input=True,
   input_device_index=RESPEAKER_INDEX,)
print("* in registrazione")
```

Lo schema del **ReSpeaker 2-Mics Pi HAT** di cui trovate i dettagli all'indirizzo **https://bit.ly/3gpG4Ym**

```
pi@raspberrypi: ~/voice-recognizer-raspi
File  Edit  Tabs  Help
  GNU nano 3.2            /home/pi/.asoundrc

pcm.!default {
        type asym
        playback.pcm {
                type plug
                slave.pcm "output"
        }
        capture.pcm {
                type plug
                slave.pcm "input"
        }
}

pcm.output {
        type hw
        card 1
}

ctl.!default {
        type hw
        card 2
}

pcm.input {
        type hw
        card 2
}
```

Ogni volta che si modificano le preferenze della **Pi** o le impostazioni del dispositivo audio, questo file va controllato. Inoltre, il volume di uscita può essere modificato nello stesso menu per la **scheda audio 1 - Cuffie** (**Jack** audio della Pi)

```
frames = []
for i in range(0, int(RESPEAKER_RATE / CHUNK *
RECORD_SECONDS)):
   data = stream.read(CHUNK)
   frames.append(data)
print("* fine registrazione")
stream.stop_stream()
stream.close()
p.terminate()
wf = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
wf.setnchannels(RESPEAKER_CHANNELS)
wf.setsampwidth(p.get_sample_size(p.get_format_from_
width(RESPEAKER_WIDTH)))
wf.setframerate(RESPEAKER_RATE)
wf.writeframes(b''.join(frames))
 wf.close()
```

Nella cartella in cui è stato eseguito lo script comparirà il file **record_test.wav**. Per verificare l'audio registrato, eseguite lo script che segue:

```
import pyaudio
import wave
CHUNK = 1024
WAVE_INPUT_FILENAME = "record_test.wav"
print("Riproduzione di un file wave: {}".format(WAVE_
INPUT_FILENAME))
wf = wave.open(WAVE_INPUT_FILENAME, 'rb')
p = pyaudio.PyAudio()
stream = p.open(format=p.get_format_from_width(wf.
getsampwidth()),
```

TIP
RPi.GPIO fornisce un modulo Python per controllare il GPIO della Raspberry Pi. Python Spidev ne offre invece uno per interfacciarsi con i dispositivi SPI dallo spazio utente tramite il driver spidev del kernel Linux. SPI (Serial Peripheral Interface) è un sistema di comunicazione seriale tra un micro controllore e altri circuiti integrati o tra più micro controllori.

```
    channels=wf.getnchannels(),
    rate=wf.getframerate(),
    output=True)
data = wf.readframes(CHUNK)
while data != b'':
    stream.write(data)
    data = wf.readframes(CHUNK)
stream.stop_stream()
stream.close()
p.terminate()
```

Prima di iniziare a preparare la Pi per emulare l'Assistente Google, esaminiamo il pulsante e i **LED RGB** presenti sul Pi HAT ReSpeaker 2-Mics.

Il pulsante del Pi HAT ReSpeaker 2-Mics

C'è un pulsante utente collegato al pin GPIO 17, come mostrato nello schema elettrico riportato in basso a destra in questa pagina. Cercheremo ora di rilevarlo con Python e la libreria **RPi.GPIO**. Prima di tutto installatela con:

```
sudo pip install rpi.gpio
```

Eseguite quindi questo script per testare il pulsante:

```
import RPi.GPIO as GPIO
import time
BUTTON = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(BUTTON, GPIO.IN)
while True:
    try:
        state = GPIO.input(BUTTON)
        if state:
            print("off")
        else:
            print("on")
        time.sleep(1)
    except KeyboardInterrupt:
        break
```

Nel terminale, potrete vedere quando viene premuto il pulsante.

```
button_test.py
```

I LED RGB integrati (APA102)

Ciascuno dei tre **LED RGB APA102** integrati nella scheda HAT è dotato di un **chip driver** aggiuntivo che si occupa di ricevere il colore desiderato tramite le sue **linee di input** e di mantenerlo fino alla ricezione di un nuovo comando. Installate la libreria **spidev**

```
sudo pip install spidev
```

e clonate il file seguente:

```
git clone https://github.com/respeaker/mic_hat.git
```

Nella sottocartella **mic_hat**, eseguite lo script **pixels.py** che farà un test su tutti i LED. Utilizzate **Ctrl + C** per fermare i LED.

```
python3 pixels.py
```

La classe **pixels** ha diverse funzioni predefinite come **pixels.wakeup()** che accende tutti i LED, **pixel.off()** che li spegne tutti, **pixels.think()** che attiva i LED uno per uno cambiando i colori e **pixels.speak()** che accende tutti i LED, modificandone l'intensità luminosa. Per utilizzare in modo efficiente le funzioni nel codice finale è necessario che i due file **apa102.py** e **pixels.py** si trovino nella stessa cartella.

In questo modo si può usare del semplice codice come quello che segue per accedere ai LED:

```
from pixels import Pixels
pixels = Pixels()
pixels.wakeup()
pixels.off()
pixels.think()
pixels.off()
pixels.speak()
pixels.off()
```

Installare i servizi di Google Voice

A questo punto tutto l'hardware (ingresso audio, uscita audio, pulsante e LED) funziona correttamente. È il momento di installare i servizi di **Google Voice** sulla Raspberry Pi. A tale scopo, seguiremo i consigli dell'eccellente tutorial all'indirizzo **https://bit.ly/3DiaDYQ**. Installate le API dell'Assistente Google, andate nella **cartella Home** della Pi e **clonate VoiceKit GitHub** nella sottocartella **/voice-recognizer-raspi**:

```
cd ~
git clone -b voicekit https://github.com/google
aiyprojects-raspbian.git ~/voice-recognizer-raspi
```

Andate nella sottodirectory appena creata ed eseguite lo script per installare i file:

```
cd ~/voice-recognizer-raspi
scripts/install-deps.sh
```

Per utilizzare **Google Voicekit** è necessario disporre di **Pico TTS** (**https://github.com/naggety/picotts**), una piattaforma di sintesi vocale che utilizza un motore **Text-to-Speech offline** per leggere i testi con voci dal suono naturale. Per installare la libreria su un sistema **Debian** procedete come segue:

```
sudo apt-get install libttspico-utils
```

Su alcune versioni di **Raspbian** questo pacchetto manca ma si può copiare il pacchetto **arm deb** da **Debian**. Su **Debian Buster**, invece, è assente, quindi per installarlo si possono usare i seguenti comandi:

```
wget http://ftp.us.debian.org/debian/pool/non-free/s/
svox/libttspico0_1.0+git20130326-9_armhf.deb
wget http://ftp.us.debian.org/debian/pool/non-free/s/
svox/libttspico-utils_1.0+git20130326-9_armhf.deb
```

Sul **ReSpeaker 2-Mics Pi HAT** c'è un pulsante utente collegato al pin **GPIO 17** come mostrato in questo schema

Ciascuno dei tre **LED RGB APA102** presenti nella scheda **ReSpeaker 2-Mics Pi HAT** è dotato di un **chip driver** aggiuntivo che si occupa di ricevere il colore desiderato tramite le linee di ingresso e di mantenerlo attivo fino alla ricezione di un nuovo comando

```
sudo apt-get install -f ./libttspico0_1.0+git20130326-9_armhf.deb ./libttspico-utils_1.0+git20130326-9_armhf.deb
```

Installate **gRPC** (**https://grpc.io/docs/what-is-grpc/introduction/**). È un moderno **framework** Open Source ad alte prestazioni per le **chiamate di procedura remota** (**RPC** da **Remote Procedure Call**). I suoi principali scenari di utilizzo sono **sistemi distribuiti** a **bassa latenza** e altamente scalabili, lo sviluppo di **client** mobili che comunicano con un server cloud e la progettazione di nuovi protocolli che devono essere accurati, efficienti e indipendenti dalla lingua. Inoltre troviamo anche e la progettazione a livelli per consentire estensioni come l'autenticazione, il bilanciamento del carico, la registrazione, il monitoraggio, ecc.

```
sudo pip install grpcio
sudo pip install grpcio-tools
```

Ora è il momento di installare le API di Google e la libreria dell'Assistente Google:

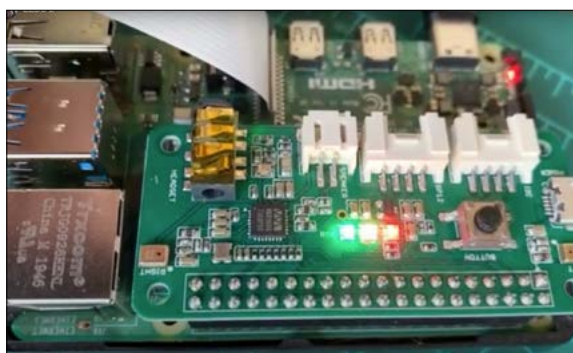
```
sudo pip install --upgrade google-api-python-client
sudo pip install --upgrade google-assistant-library==1.0.1
sudo pip install --upgrade google-assistant-sdk[samples]==0.5.1
```

A questo punto i pacchetti principali sono stati installati, quindi riavviate il sistema:

```
reboot
```

Sono necessarie delle modifiche al **kit Google Voice** originale. Aprite il file

```
/home/pi/voice-recognizer-raspi/src/aiy/_apis/_speech.py
```

e commentate le seguenti righe:

Lo **script pixels.py** nella sottocartella **mic_hat** esegue un test su tutti i **LED**. Usate **Ctrl + C** per fermarli

```
#try:
# from google.cloud import speech
# from google.cloud.speech import enums
# from google.cloud.speech import types
#except ImportError:
# print("Failed to import google.cloud.speech. Try:")
# print(" env/bin/pip install -r requirements.txt")
# sys.exit(1)c
```

Aprite ora il file che segue:

```
/home/pi/voice-recognizer-raspi/src/aiy/voicehat.py
```

Sostituite a questo punto il pin GPIO del pulsante (che appare come **23**) con quello utilizzato sul ReSpeaker 2-Mics Pi HAT (che è il **17**).

```
_GPIO_BUTTON = 17
```

A questo punto, tutti i componenti hardware e software dovrebbero essere completi. La parte mancante è ottenere le credenziali da Google per eseguire il **Voice Kit** sulla vostra Raspberry Pi.

Abilitare l'API dell'Assistente Google

Tutti i passaggi per l'abilitazione dell'API sono riportati sul sito di **Google AIY Voice Kit** (**https://bit.ly/3CP6qKJ**) nella sezione **GET CREDENTIALS** che si trova in fondo alla pagina. Bisogna quindi aprire una seconda pagina **https://console.cloud.google.com/** (la pagina principale della piattaforma cloud di Google) e seguire le istruzioni. Se la procedura è eseguita correttamente, viene scaricato un file **JSON** sul computer. Le istruzioni spiegano anche come creare un file **assistant.json** in cui salvare il contenuto del file scaricato. Va poi salvato nella cartella **Home** della Pi. Oltre a seguire le istruzioni citate, è importante anche inserire la propria email in **Test users**, utilizzando le opzioni a cui si accede facendo click su **+ ADD USERS** nella pagina di consenso di **OAuth**.

E questo è quanto: è il momento di testare il vostro assistente vocale! Andate nella sottocartella in cui è stato installato il Voice Kit:

```
cd ~/voice-recognizer-raspi
```

Eseguite quindi il programma **demo** che si trova nella sottocartella **/src**:

```
python3 src/assistant_grpc_demo.py
```

Se tutto è corretto, dovreste ottenere i seguenti messaggi sul terminale:

TIP

Il ReSpeaker 2-Mics Pi HAT di Seeed Studio è un'interfaccia utente vocale per Raspberry Pi. Offre 2 microfoni analogici e il codec audio WM8960 per l'acquisizione della voce ad alta definizione.

> **TIP**
>
> **Una chiamata di procedura remota o RPC è l'attivazione da parte di un programma di una procedura o subroutine su un computer diverso da quello su cui viene eseguito. L'RPC consente quindi a un programma di eseguire subroutine "a distanza" su computer remoti attraverso una rete.**

```
pi@raspberrypi:~/voice-recognizer-raspi $ python3 src/
assistant_grpc_demo.py /home/pi/voice-recognizer-raspi/
src/aiy/_drivers/_led.py:51: RuntimeWarning: This
channel is already in use, continuing anyway. Use GPIO.
setwarnings (False) to disable warnings. GPIO.
setup(channel, GPIO.OUT)
Press the button and speak
[2021-01-22 16:39:34,571] INFO: recorder:started recording
```

L'assistente vocale è ora in attesa che premiate il pulsante per avviare una conversazione.

Includere i LED dell'HAT nel codice

Come ultima parte del progetto, includiamo i LED presenti sull'HAT nel codice precedente, come mostrato di seguito. Copiate i file **apa102.py** e **pixels.py** nella stessa directory in cui eseguite il codice sottostante (in questo caso si tratta di **voice-recognizer-raspi/src**)

```
import time
import aiy.assistant.grpc
import aiy.audio
import aiy.voicehat
from pixels import Pixels
import logging
pixels = Pixels()
pixels.off()
logging.basicConfig(
   level=logging.INFO,
   format="[%(asctime)s] %(levelname)s:%(name)
s:%(message)s"
)
def wakeup_assistant():
   pixels.wakeup()
   pixels.think()
   time.sleep(3)
   pixels.speak()
   time.sleep(3)
   pixels.off()
def main():
   wakeup_assistant()
   status_ui = aiy.voicehat.get_status_ui()
   status_ui.status('starting')
   assistant = aiy.assistant.grpc.get_assistant()
   button = aiy.voicehat.get_button()
with aiy.audio.get_recorder():
      while True:
         pixels.off()
         status_ui.status('ready')
         print('Premi il pulsante e parla')
         button.wait_for_press()
         status_ui.status('listening')
         print('In ascolto...')
         pixels.think()
         text, audio = assistant.recognize()
         if text:
            if text == 'goodbye':
               status_ui.status('stopping')
               print('Ciao!')
               pixels.off()
               time.sleep(1)
               break
            print('Hai detto "', text, '"')
         if audio:
            pixels.speak()
            aiy.audio.play_audio(audio)
            pixels.off()
   pixels.off()
if __name__ == '__main__':
main()
```

Oltre a seguire le istruzioni citate, è importante anche inserire la propria email in **Test users** utilizzando le opzioni **+ ADD USERS** nella pagina di consenso di **OAuth**

Ora, durante il processo di avvio, verrà aggiunta (una sola volta) una sorta di dimostrazione dei LED. Inoltre, ogni volta che premete il pulsante, l'assistente vocale "pensa" in attesa della vostra domanda. Per questo useremo la funzione **pixels.think()** che costringe i LED a scorrere. Lo stesso vale quando l'assistente sta "parlando": i LED manterranno in questo caso il loro colore **RGB** ma sfumeranno.

Giocare con i pin GPIO

Un vantaggio significativo di emulare l'Assistente Google su una Pi è che si possono usare i suoi **pin GPIO** per controllare oggetti esterni nel mondo reale.

» MICRO APPRENDIMENTO AUTOMATICO

TinyML consente all'intelligenza artificiale di essere vicina al mondo fisico e l'esecuzione di modelli di apprendimento automatico (**ML** da **machine learning**) a livello di microprocessore evita problemi di **latenza**, consumo energetico e sicurezza. Secondo **Tinyml.org**: "Il **Tiny Machine Learning** è definito in senso lato come un campo in rapida crescita di tecnologie e applicazioni di apprendimento automatico che includono hardware, algoritmi e software in grado di eseguire analisi dei dati dei sensori su dispositivi a bassissima potenza (tipicamente nell'ordine dei **mW** e inferiori) e quindi di consentire una varietà di casi d'uso sempre attivi e di puntare a dispositivi alimentati a batteria". Per saperne di più sul TinyML un ottimo libro è **TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers** (**https://amzn.to/3gp83qZ**).

Lo abbiamo già fatto utilizzando i LED e il pulsante di ReSpeaker 2-Mics Pi HAT. L'HAT lascia disponibili due pin GPIO (**12** e **13**) attraverso il **connettore Grove**, come illustrato nel suo schema elettrico riportato in basso in questa pagina. Installiamo ora un pulsante esterno sul **pin 13** del GPIO della Pi e un LED sul **pin 12**, come mostrato nel diagramma qui accanto. Modifichiamo lo script utilizzato in precedenza per testare il pulsante HAT per provare ora il tasto esterno e il LED che si accenderà ogni volta che il pulsante verrà premuto:

```
import RPi.GPIO as GPIO
import time
BUTTON = 13
LED = 12
GPIO.setmode(GPIO.BCM)
GPIO.setup(BUTTON, GPIO.IN)
GPIO.setup(LED, GPIO.OUT)
GPIO.output(LED, GPIO.LOW)
while True:
    try:
        state = GPIO.input(BUTTON)
        if state:
            GPIO.output(LED,GPIO.LOW)
            print("off")
        else:
            GPIO.output(LED,GPIO.HIGH)
            print("on")
        time.sleep(1)
    except KeyboardInterrupt:
        GPIO.cleanup()
        break
print("pulisci")
GPIO.cleanup() # pulisce tutto il GPIO
```

Per risvegliare l'assistente vocale usando un pulsante esterno, l'unica cosa da fare è modificare il pin GPIO del pulsante dell'assistente. Aprite il file:

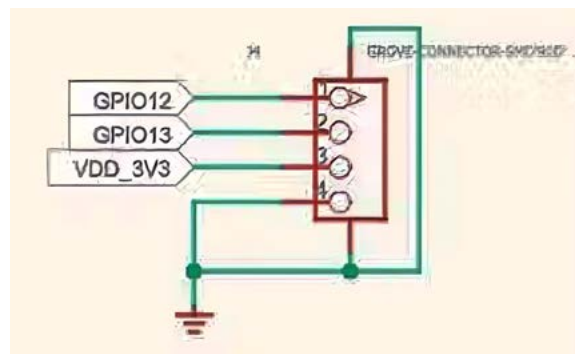
```
/home/pi/voice-recognizer-raspi/src/aiy/voicehat.py
```

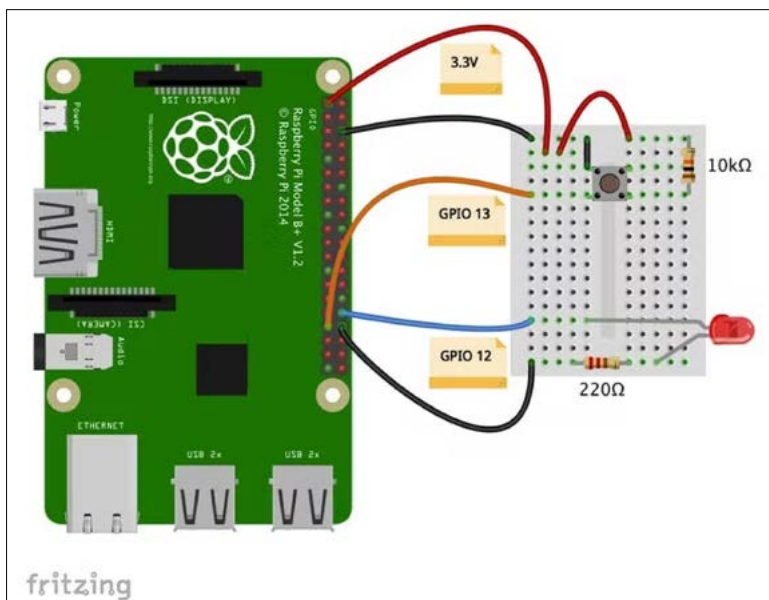
Sostituite quindi il pin GPIO del pulsante (**17**) con quello utilizzato dal tasto esterno (**13**).

```
_GPIO_BUTTON = 13
```

D'ora in poi, ogni volta che si premerà il pulsante esterno, l'assistente vocale si "sveglierà".

Controllo vocale di un dispositivo esterno

Modifichiamo anche il codice completo utilizzato in precedenza per incorporare anche il LED che dovrebbe ricevere alcuni comandi vocali come

L'**HAT** lascia liberi **2 pin GPIO** (**12** e **13**) attraverso il **connettore Grove**

Lo schema per collegare un pulsante esterno al **pin 13** del **GPIO** della **Pi** e un **LED** al pin **12**

l'accensione o lo spegnimento. Qui sotto viene riportato il codice completo:

```
import time
import aiy.assistant.grpc
import aiy.audio
import aiy.voicehat
from pixels import Pixels
import logging
import RPi.GPIO as GPIO
LED = 12
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED, GPIO.OUT)
GPIO.output(LED, GPIO.LOW)
pixels = Pixels()
pixels.off()
logging.basicConfig(
    level=logging.INFO,
    format="[%(asctime)s] %(levelname)s:%(name)
s:%(message)s"
)
def led_blink():
    for i in range (0,6):
    GPIO.output(LED,GPIO.HIGH)
    time.sleep(0.25)
    GPIO.output(LED,GPIO.LOW)
    time.sleep(0.25)
def wakeup_assistant():
    pixels.wakeup()
    pixels.think()
    time.sleep(3)
    pixels.speak()
    time.sleep(3)
    pixels.off()
def main():
    wakeup_assistant()
    status_ui = aiy.voicehat.get_status_ui()
    status_ui.status('avvio in corso')
    assistant = aiy.assistant.grpc.get_assistant()
    button = aiy.voicehat.get_button()
with aiy.audio.get_recorder():
```

Il pulsante esterno e il **LED** collegati

```
    while True:
        play_audio = True
        pixels.off()
        status_ui.status('ready')
        print('Premi il pulsante e parla')
        button.wait_for_press()
        status_ui.status('listening')
        print('In ascolto...')
        pixels.think()
        text, audio = assistant.recognize()
if text:
            if text == 'goodbye':
                status_ui.status('stopping')
                print('Ciao!')
                pixels.off()
                time.sleep(1)
                break
            if 'turn on' in text:
                pixels.off()
                GPIO.output(LED,GPIO.HIGH)
                play_audio = False
            if 'turn off' in text:
                pixels.off()
                GPIO.output(LED,GPIO.LOW)
                play_audio = False
            if 'blink' in text:
                pixels.off()
                led_blink()
                play_audio = False
            print('Hai detto "', text, '"')
if play_audio:
            if audio:
                pixels.speak()
                aiy.audio.play_audio(audio)
                pixels.off()
    pixels.off()
if __name__ == '__main__':
main()
```

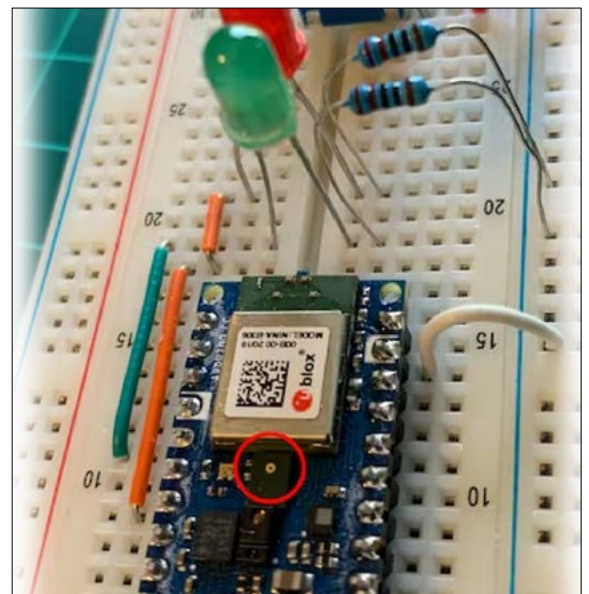
TIP

L'inferenza nel machine learning è il processo di esecuzione dei punti di dati in un modello di apprendimento automatico per calcolare un risultato, per esempio un singolo punteggio numerico. Questo processo si può descrivere anche come messa in produzione di un modello di apprendimento automatico.

Utilizzo del Keyword Spotting (KWS)

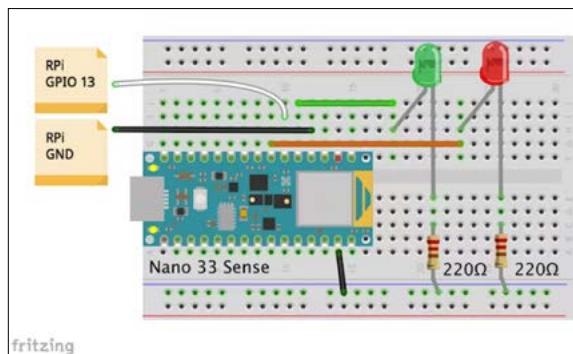
Finora il metodo usato per risvegliare il nostro assistente vocale è stato un pulsante fisico ma, come discusso nell'introduzione, gli assistenti vocali come Google Home dovrebbero reagire quando vengono utilizzate particolari parole chiave o **wake word** come "Hey Google".
Ora sostituiremo il pulsante fisico con uno "virtuale", impiegando un meccanismo noto come **KWS**, o **Keyword Spotting** (individuazione delle parole chiave). Utilizzeremo un Arduino Nano 33 BLE Sense: è un microcontrollore che ha incorporato un microfono digitale che verrà utilizzato per individuare la parola chiave.
Per semplificare (visto che l'oggetto principale di questo progetto non è lo sviluppo di modelli di **reti neurali**), sfruttiamo del codice sviluppato da Google incorporato in **Arduino IDE**, creato con **TensorFlowLite** e denominato **micro_speech**. Questo **sketch** incorpora un modello in grado di individuare le due parole **yes** e **no**, oltre a una parola non nota e al silenzio. È possibile ottenere lo sketch da **Arduino IDE > Files > Examples > Arduino TensorFlowLite > micro_speech**.
Per la prova, potete caricare il codice così com'è nel vostro Arduino Nano e testarlo dicendo "yes" o "no". Quando viene riconosciuto il termine, il LED RGB interno si accende (**yes** corrisponde al verde e **no** al rosso).

Fare il riconoscimento vocale su un microprocessore

Arduino eseguirà l'**inferenza** in un modello pre-addestrato sviluppato con TensorFlow. Un modello di **rete neurale convoluzionale o CNN** (**tiny_conv**) viene addestrato con oltre **100.000** campioni di registrazione di un secondo (o meno) (formato **.wave**) di **35 fonemi** diversi

Il **microcontrollore Arduino Nano 33 BLE Sense** ha incorporato un microfono digitale che qui viene utilizzato per individuare la parola chiave

(**Google Speech Commands Dataset**). Il modello addestrato viene convertito in un **array di byte in C** da **TensorFlowLite** per essere utilizzato su un microprocessore di piccole dimensioni come il **Nano** (il modello finale ha un'accuratezza superiore al **90%** e delle dimensioni di soli **19 KB**). Si noti che i dati grezzi (suono in formato **.wave**) non possono essere utilizzati direttamente con un modello CNN. Per prima cosa, i dati sonori devono essere convertiti in un'immagine (**40 x 49 pixel**), il che viene fatto utilizzando un **MFCC Feature Converter**. Vediamo come funziona il codice in termini generali... Arduino resta continuamente in ascolto del flusso sonoro che lo circonda. Il microfono cattura campioni audio (da un secondo) e li converte in dati a **16 bit** (**modulazione a impulsi codificati** o **PCM** da **Pulse Code Modulation**) attraverso il modulo **Audio Provider**. I dati PCM devono essere pre-elaborati prima di essere utilizzati come inferenza. Nel modulo **Feature Provider**, i dati grezzi vengono convertiti in immagini dall'MFCC Feature Converter. Ogni campione sarà un'immagine monocromatica (o un **tensore** di dimensioni: **1, 49, 40, 1**). L'**interprete TFLite** esegue l'inferenza, o meglio, classifica il tensore in ingresso in quattro classi distinte. L'output sarà un tensore di dimensione **1, 4** in cui i valori sono le probabilità che il suono in ingresso sia silenzio, sconosciuto, yes o no. In base a queste probabilità, il modulo **Command Recognizer & Responder** utilizzerà l'output dell'interprete TFLite per decidere se un comando è stato ricevuto e intraprendere le azioni appropriate. Per esempio, se il suono ha una maggiore probabilità di essere un yes, si accenderà il LED RGB interno verde, se è un no, si attiverà quello rosso e infine, per qualsiasi altra parola possibile, si illuminerà il blu. Il modello è **quantizzato** in modo da funzionare con **numeri interi a 8 bit**, quindi la probabilità va da **0** a **255**. Se la probabilità è superiore a **200** (circa l'**80%**) il comando viene eseguito.

Modificare hardware e codice

Installeremo due LED esterni per replicare quanto fatto con il LED RGB interno del Nano. Il LED verde sarà collegato all'uscita **D2** del Nano e quello rosso alla **D4**. L'uscita **D3** del Nano (insieme a **GND**) sostituirà il pulsante fisico

L'uscita **D3** del Nano (insieme a **GND**) sostituisce il pulsante fisico esterno collegato al **pin 13** del **GPIO** della **Pi**

esterno collegato al **pin 13** del GPIO della Pi. Dobbiamo modificare il codice in modo che ogni volta che viene individuata la parola "yes" (la nostra wake word), il LED verde esterno si accenda e venga inviato un impulso alla Pi, simulando la pressione di un pulsante. Come si può vedere, l'unico modulo che deve essere modificato è Command Recognizer & Responder. Questo codice si trova nella scheda: **arduino_command_responder.cpp**. Di seguito, le nuove parti di codice che devono esservi aggiunte:

```
...
#define GREEN_LED_PIN 2
#define SIM_BUT_PIN 3
#define RED_LED_PIN 4
// Crea una funzione per simulare un pulsante premuto per
500ms
void buttonSimulator(){
  pinMode(SIM_BUT_PIN, OUTPUT);
  digitalWrite(SIM_BUT_PIN, LOW);
  delay(500);
  pinMode(SIM_BUT_PIN, INPUT);
}
...
pinMode(GREEN_LED_PIN, OUTPUT);
pinMode(RED_LED_PIN, OUTPUT);
pinMode(SIM_BUT_PIN, INPUT); // Stato aperto
...
// Se viene percepito un comando, accende i LED
appropriati e invia un segnale alla Pi
    if (found_command[0] == 'y') {
      last_command_time = current_time;
      digitalWrite(LEDG, LOW);  // Verde per yes
      digitalWrite(GREEN_LED_PIN, HIGH);  // HIGH per yes
      buttonSimulator(); // Simula il pulsante
    }
if (found_command[0] == 'n') {
      last_command_time = current_time;
      digitalWrite(LEDR, LOW);  // Rosso per no
      digitalWrite(RED_LED_PIN, HIGH);  // HIGH per yes
    }
...
// Se last_command_time è diverso da zero ma risale a >3
secondi fa, azzera
  // e spegne il LED.
  if (last_command_time != 0) {
    if (last_command_time < (current_time - 3000)) {
      last_command_time = 0;
      digitalWrite(LED_BUILTIN, LOW);
      digitalWrite(GREEN_LED_PIN, LOW);
      digitalWrite(RED_LED_PIN, LOW);
      digitalWrite(LEDR, HIGH);
      digitalWrite(LEDG, HIGH);
      digitalWrite(LEDB, HIGH);
    }
...
```

E questo è tutto! Ora sapete come si crea un assistente vocale intelligente. Potete vedere il progetto finito in azione nel video che si trova nell'introduzione dell'articolo all'indirizzo **https://bit.ly/3TEWbiR**, dove sono disponibili anche i file aggiornati e il codice Arduino completo. LXP

TIP

I coefficienti cepstrali di frequenza mel (MFCC) sono coefficienti che compongono collettivamente un MFC (cepstrum frequenza mel), cioè una rappresentazione dello spettro di potenza a breve termine di un suono, basata su una trasformazione lineare del coseno di uno spettro di potenza log su una scala mel non lineare di frequenza.

Crediti: Tomas Petricek

Create il vostro Excel in 100 righe di F#

Un linguaggio di programmazione funzionale e succinto vi permette di scrivere un foglio di calcolo in modo veloce ed efficiente

Produrre una semplice **applicazione Web** di un **foglio di calcolo** simile a **Excel** in **F#** non è troppo difficile e dimostra tutte le grandi caratteristiche di questo linguaggio, come la modellazione del **dominio** con i **tipi**, la potenza della **composizionalità** e anche come l'**approccio funzionale** possa essere incredibilmente potente per la creazione di interfacce utente. Il modo migliore per spiegare ciò che vogliamo costruire è darvi una demo dal vivo con cui potete giocare e l'autore di questo progetto ha già implementato sia i **numeri di Fibonacci** (**colonna B**) sia quelli fattoriali (**colonna D**) nel foglio di calcolo. Potete eseguire e modificare il codice nel vostro browser usando il **REPL** di **Fable** (**https://fable.io/repl/**). Si trova in **Samples** > **UI** > **Spreadsheet**. Si può fare click su qualsiasi cella per modificarla. Per confermare la modifica basta fare click su qualsiasi altra cella. È possibile inserire numeri come **1** (nella cella **B1**) o formule come **=B1+B2** nella cella **B3**. Le formule supportano le parentesi e quattro **operatori numerici** standard. Quando si effettua una

	A	B	C	D	E	F	G	H
1		1		1				
2		1		2				
3		2		6				
4		3		24				
5		5		120				
6		8		720				
7		13		5040				
8		21		40320				

Nel foglio di calcolo sono già implementati sia i **numeri di Fibonacci** (**colonna B**) sia quelli **fattoriali** (**colonna D**) e potete provarlo online

```
1:  type Position = char * int
2:
3:  type Expr =
4:    | Number of int
5:    | Reference of Position
6:    | Binary of Expr * char * Expr
7:
8:  type Sheet = Map<Position, string>
```

Per le posizioni e le espressioni del foglio di calcolo e lo **spreadsheet** stesso definiamo i tipi per **Position**, **Expr** e **Sheet**

modifica, il foglio di calcolo si aggiorna automaticamente. Se si commette un errore di sintassi, si fa riferimento a una cella vuota o si crea un **riferimento ricorsivo**, il foglio di calcolo mostra **#ERR**.

Definire il modello di dominio

Seguendo il tipico stile di sviluppo guidato dai tipi di F#, la prima cosa a cui bisogna pensare è il **modello di dominio**. I tipi in questo caso devono rappresentare ciò con cui si lavora in un'applicazione di fogli di calcolo. Ci sono posizioni come **A5** o **C10**, espressioni come **=A1+3** e il foglio stesso che contiene gli **input** dell'utente in alcune **celle**. Per modellare questi elementi, definiamo i tipi per `Position`, `Expr` e `Sheet`:

```
type Position = char * int

type Expr =
 | Number of int
 | Reference of Position
 | Binary of Expr * char * Expr

type Sheet = Map<Position, string>
```

Una posizione è semplicemente una coppia composta da un nome di colonna e da un numero. Un'espressione è più interessante, perché è ricorsiva. Per esempio, **A1+3** è un'applicazione di un operatore binario alle **sottoespressioni A1**,

Potete eseguire e modificare il codice nel vostro browser usando il **REPL** di **Fable** (**https://fable.io/repl/**). Si trova in **Samples > UI > Spreadsheet**

(ossia un riferimento) e **3**, una costante numerica. In F#, questo aspetto è ben rappresentato da un'**unione discriminata**. Nel caso binario, le sottoespressioni sinistra e destra sono a loro volta valori del tipo `Expr`, quindi il nostro tipo `Expr` è ricorsivo. Il tipo `Sheet` è una mappa che va dalle posizioni agli input grezzi dell'utente. Potremmo anche memorizzare **espressioni parsate** o persino **risultati valutati**, ma abbiamo sempre bisogno dell'input originale in modo che l'utente possa modificarlo. Per semplificare, memorizzeremo l'input originale e lo analizzeremo ogni volta che dovremo valutare il valore di una cella. Per eseguire il **parsing** e la **valutazione**, definiremo in seguito due funzioni:

```
val parse : string -> Expr option
val evaluate : Expr * Sheet -> int option
```

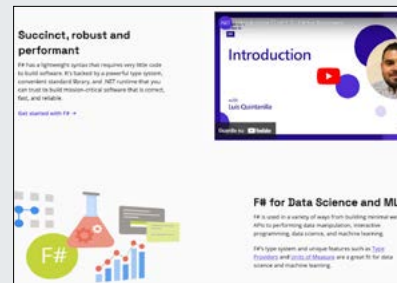
Le tratteremo più avanti, quando discuteremo la logica del nostro foglio di calcolo, ma scrivere il tipo in anticipo è utile. Con questi tipi possiamo già vedere come tutto si incastra. Data una posizione, possiamo fare una ricerca in `Sheet` per trovare il testo inserito, poi analizzarlo utilizzando `parse` per ottenere `Expr` e, infine, passare l'espressione a `evaluate` per ottenere il valore risultante. Vediamo anche che sia `parse` sia `evaluate` possono fallire. Il primo se l'input non è una formula valida e il secondo se si fa riferimento a una cella vuota. Ora non resta che continuare a scrivere il resto del nostro **Excel** finché il controllore di tipo non è soddisfatto!

Creare l'interfaccia utente con Elmish

Tratteremo l'interfaccia utente e poi torneremo sull'implementazione della logica di parsing e di valutazione. Per creare interfacce utente, Fable è dotato di un'ottima libreria chiamata **Elmish**. Implementa un'architettura funzionale dell'interfaccia utente resa popolare dal linguaggio **Elm**, nota anche come **model view update**. L'idea dell'architettura è estremamente semplice. Sono sufficienti i seguenti due tipi e due funzioni:

```
type State = (Record capturing the state)
type Event = (Union listing possible events)

val update : State -> Event -> State
val view : State -> (Event -> unit) -> Html
```

» UN LINGUAGGIO DA SCOPRIRE

F#, che si pronuncia come la parola inglese **F Sharp** (**Fa diesis** in italiano), è un linguaggio Open Source il cui principale contributore è **Microsoft**. Il luogo di incontro e punto di riferimento della comunità è però l'indipendente **F# Software Foundation** (**https://fsharp.org/**), sul cui sito sono pubblicate documentazione, guide ed esercitazioni. Esplorando le sue pagine potrete trovare molti spunti utili per conoscere meglio questo linguaggio di programmazione multi-paradigma, basato su **.NET Framework** (una piattaforma Open Source per sviluppatori, anch'essa creata da Microsoft), che permette la **programmazione funzionale** così come quella **imperativa** e quella **a oggetti**. Nelle parole della Foundation: "F# offre la semplicità e la sinteticità di **Python** con la correttezza, la robustezza e le prestazioni di **C#** o **Java**". Ha una sintassi leggera che richiede la scrittura di pochissimo codice per realizzare il software. F# fa parte della famiglia dei linguaggi **ML** ed è nato come implementazione in .NET Framework di un nucleo del linguaggio di programmazione **OCaml**. È stato influenzato anche da **C#**, **Python**, **Haskell**, **Scala** ed **Erlang**. Per lavorare con F# in **Linux** sono disponibili numerosi strumenti professionali, tra cui **Visual Studio Code**. È possibile trovarli su **https://bit.ly/3Ebit6Y**.

Succinto e dalle ottime prestazioni, **F#** è utilizzato in molti campi, tra cui l'apprendimento automatico

I due tipi e le due funzioni definiscono l'interfaccia utente come segue:
- `State` memorizza la parte dello stato dell'interfaccia utente che serve per disegnarla;
- `Event` è un'unione di diversi eventi che possono verificarsi quando l'utente interagisce con l'interfaccia dell'applicazione;
- `update` è una funzione che prende uno stato originale e un evento e produce di conseguenza un nuovo stato modificato;
- `view` prende lo stato e genera un documento **HTML**. Prende anche una funzione `Event -> unit` che può essere utilizzata nei gestori di eventi del documento HTML per attivare un evento.

Concettualmente, si può pensare che l'applicazione parta con uno stato iniziale, **renderizzi** una pagina e, quando avviene un'azione o un evento, aggiorni lo stato usando `update` e ridisegni la pagina usando `view`. L'elemento chiave di questo approccio è che Elmish non sostituisce l'intero **DOM**, ma confronta il nuovo documento con l'ultimo e aggiorna solo gli elementi del DOM che sono cambiati. Quali stati ed eventi sono però presenti nel foglio di calcolo? Come per l'intera applicazione, il primo passo per implementare l'interfaccia utente è la definizione di alcuni tipi:

TIP

Un modello di dominio è un modello concettuale di un sistema. Descrive le varie entità che ne fanno parte e le loro relazioni. I tipi si possono usare per rappresentare il dominio in modo dettagliato e, in molti casi, consentono di codificare le regole, in modo che non sia possibile creare codice errato.

TIP

Fable (https://fable.io/) è un compilatore che porta F# nell'ecosistema JavaScript. Genera un output JavaScript moderno, interagisce con i pacchetti del linguaggio e supporta diversi modelli di sviluppo, tra cui React.

```
type Event =
  | UpdateValue of Position * string
  | StartEdit of Position

type State =
  { Rows : int list
    Cols : char list
    Active : Position option
    Cells : Sheet }
```

Nello stato sono conservati un elenco di chiavi di riga e di colonna (in genere inizia da **A1**, ma non è necessario che sia così), la cella attualmente selezionata (può essere **None** se non è selezionata alcuna cella) e, infine, le celle del foglio di calcolo. Ci sono due tipi di eventi. `UpdateValue` si verifica quando si modifica il testo nella cella corrente mentre l'evento `StartEdit` si produce quando fate click su un'altra cella per iniziare a modificarla.

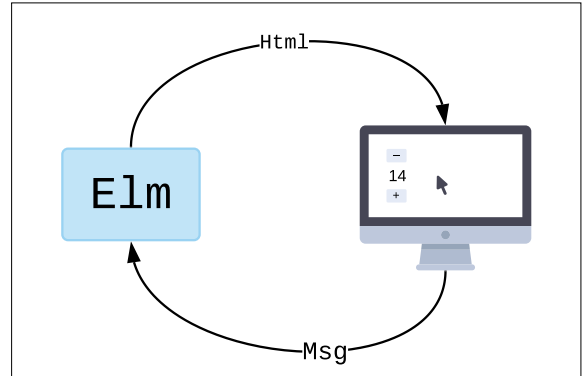
Aggiornare e renderizzare il foglio

In Elmish, la funzione di aggiornamento è un po' più complicata di quanto detto sopra. Oltre a un nuovo stato, possiamo anche restituire un elenco di comandi. Questi ultimi sono usati per dire al sistema che deve iniziare un'azione dopo l'aggiornamento dello stato. Può trattarsi, per esempio, di avviare una **richiesta HTTP** per recuperare alcune informazioni dal server. Nel nostro caso non abbiamo bisogno di alcun comando quindi restituiamo semplicemente `Cmd.none` :

```
let update msg state =
  match msg with
  | StartEdit(pos) ->
      { state with Active = Some pos }, Cmd.none

  | UpdateValue(pos, value) ->
      let newCells = Map.add pos value state.Cells
      { state with Cells = newCells }, Cmd.none
```

Per costruire il documento HTML, Elmish fornisce un **wrapper** leggero costruito sulla base di **React** (sebbene sia possibile utilizzare anche altre **librerie DOM virtuali**). Definisce funzioni **tipizzate** per creare gli elementi HTML e specificarne gli attributi. Implementeremo prima la funzione di visualizzazione principale (che genera la **griglia** del nostro foglio di calcolo)

```
1:  let view state trigger =
2:    table [] [
3:      thead [] [
4:        tr [] [
5:          yield th [] []
6:          for col in state.Cols -> th [] [ str (string col) ]
7:        ]
8:      ]
9:      tbody [] [
10:       for row in state.Rows -> tr [] [
11:         yield th [] [ str (string row) ]
12:         for col in state.Cols -> renderCell trigger (col, row) state
13:       ]
14:     ]
15:   ]
```

In questo caso, stiamo usando le **comprensioni di lista** di **F#** per creare il documento **HTML**. Per esempio, le righe **4-7** generano l'intestazione della tabella

Elmish permette di scrivere applicazioni in **F#** seguendo lo stile di architettura "model view update" di **Elm**

e poi discuteremo la **funzione helper** `renderCell` che disegna una singola cella.

```
let view state trigger =
  table [] [
    thead [] [
      tr [] [
        yield th [] []
        for col in state.Cols -> th [] [ str (string col) ]
      ]
    ]
    tbody [] [
      for row in state.Rows -> tr [] [
        yield th [] [ str (string row) ]
        for col in state.Cols -> renderCell trigger (col,
row) state
      ]
    ]
  ]
```

In questo caso, stiamo usando le **comprensioni di lista** di F# per generare il documento HTML. Per esempio, le righe **4-7** generano l'intestazione della tabella. Creiamo un elemento `tr` senza attributi (il primo argomento) contenente una coppia di elementi `th` (il secondo argomento). Utilizziamo `yield` per generare gli elementi: prima creiamo quello vuoto `th` nell'angolo in alto a sinistra, poi **iteriamo** su tutte le colonne e produciamo un'intestazione per ciascuna di esse. La variabile `col` è un carattere, quindi la trasformiamo in una stringa usando `string` prima di convertirla in contenuto HTML usando la funzione `str` fornita da Elmish. Il vantaggio di scrivere il rendering HTML in questo modo è che è componibile. Non dobbiamo mettere tutto all'interno di un'unica funzione. Qui richiamiamo `renderCell` (**riga 12**) per visualizzare il contenuto di una cella.

La visualizzazione di una cella

Ci sono due modi diversi per rappresentare una cella. Per quella selezionata, bisogna visualizzare un editor con una casella di input contenente il testo inserito in origine. Per tutte le altre, è necessario analizzare la formula, valutarla e visualizzare il risultato. La funzione `renderCell` sceglie il ramo e, nel secondo caso, si occupa

```
1:  let renderView trigger pos (value:option<_>) =
2:    let color = if value.IsNone then "#ffb0b0" else "white"
3:    td
4:      [ Style [Background color]
5:        OnClick (fun _ -> trigger(StartEdit(pos)) ) ]
6:      [ str (defaultArg value "#ERR") ]
7:
8: let renderEditor trigger pos value =
9:    td [ Class "selected"] [
10:     input [
11:       AutoFocus true
12:       OnInput (fun e -> trigger(UpdateValue(pos, e.target?value)))
13:       Value value ]
14:   ]
```

In **renderEditor** e **renderView** creiamo dei gestori che attivano degli eventi quando accade qualcosa nell'interfaccia utente

anche della valutazione:

```
let renderCell trigger pos state =
  if state.Active = Some pos then
    let text = Map.tryFind pos state.Cells
    renderEditor trigger pos (defaultArg text "")
  else
    match Map.tryFind pos state.Cells with
    | Some input ->
      let result =
        parse input
        |> Option.bind (evaluate Set.empty state.Cells)
        |> Option.map string
      renderView trigger pos result
    | _ -> renderView trigger pos (Some "")
```

Verifichiamo se la cella che viene renderizzata è quella attiva utilizzando la condizione `state.Active = Some pos`. Invece di confrontare due valori di `Position`, confrontiamo quelli di `Position option` e non dobbiamo preoccuparci del caso in cui `state.Active` sia `None`. Se la cella corrente è attiva, si prende il valore inserito o la stringa vuota e la si passa a `renderEditor` (definito successivamente). In caso contrario, si tenta di ottenere l'input; se non c'è alcun input, si richiama `renderView` con `Some ""` per eseguire il rendering di una cella valida ma vuota. Altrimenti, utilizziamo una sequenza di `parse` ed `evaluate` per ottenere il risultato. Analizzeremo entrambe queste funzioni più avanti, quando discuteremo di come viene implementata la logica del foglio di calcolo. Sia `parse` sia `evaluate` possono fallire, quindi usiamo il tipo `option` per comporli. `Option.bind` esegue `evaluate` solo quando `parse` ha successo; altrimenti propaga il risultato `None`. Utilizziamo anche `Option.map` per trasformare il risultato opzionale di tipo `int` in una stringa opzionale da passare a `renderView`. Finora non sono stati creati **handler** che attivino eventi quando accade qualcosa nell'interfaccia utente. Lo faremo ora in `renderEditor` e `renderView` che per il resto sono entrambi abbastanza semplici:

```
let renderView trigger pos (value:option<_>) =
  let color = if value.IsNone then "#ffb0b0" else
"white"
  td
    [ Style [Background color]
      OnClick (fun _ -> trigger(StartEdit(pos)) ) ]
    [ str (defaultArg value "#ERR") ]

let renderEditor trigger pos value =
  td [ Class "selected"] [
    input [
      AutoFocus true
      OnInput (fun e -> trigger(UpdateValue(pos, e.
target?value)))
      Value value ]
  ]
```

In `renderView`, creiamo uno sfondo rosso e utilizziamo la stringa **#ERR** se il valore da visualizzare è vuoto (indicando un errore). Aggiungiamo anche un **gestore** `OnClick`. Quando l'utente fa click su una cella, vogliamo attivare l'evento `StartEdit` per spostare l'editor nella cella corrente. Per farlo, specifichiamo l'attributo `OnClick` e, quando si verifica un click, attiviamo l'evento utilizzando la funzione `trigger` che abbiamo ottenuto come argomento di input per la funzione `view` (e che abbiamo passato prima a `renderCell` e poi a `renderView`). La funzione `renderEditor` è simile. Specifichiamo il gestore `OnInput` e, ogni volta che il testo nell'input cambia, attiviamo l'evento `UpdateValue` per aggiornare il valore e ricalcolare tutto nel foglio di calcolo. Specifichiamo anche l'attributo `AutoFocus` che assicura che l'elemento sia attivo subito dopo la sua creazione (quando si fa click su una cella).

Assemblare tutto l'insieme

Ora abbiamo tutti i componenti necessari per gestire l'interfaccia utente. Ci sono le definizioni dei tipi `State` ed `Event` e le funzioni `update` e `view`. Per mettere tutto insieme, bisogna definire lo stato iniziale, specificare l'**ID** dell'elemento HTML in cui l'applicazione deve essere resa e avviarla.

```
let initial () =
  { Cols = ['A' .. 'K']
    Rows = [1 .. 15]
    Active = None
    Cells = Map.empty },
  Cmd.Empty

Program.mkProgram initial update view
|> Program.withReact "main"
|> Program.run
```

Lo stato iniziale definisce gli intervalli di righe e colonne disponibili e specifica che non ci sono valori in nessuna delle celle (la demo integra qui le celle iniziali per il calcolo dei numeri fattoriali e di Fibonacci). Quindi utilizziamo `mkProgram` per riunire tutti i componenti, specifichiamo React come motore di esecuzione e avviamo l'applicazione Elmish!

La logica del foglio di calcolo

Finora abbiamo definito il modello di dominio che specifica che cos'è un foglio di calcolo utilizzando i tipi di F# e abbiamo implementato l'interfaccia utente con Elmish. L'unica cosa che abbiamo saltato a questo punto è la logica del foglio di calcolo, cioè l'analisi e la valutazione delle formule. Completare queste due operazioni è più

TIP

Il Document Object Model (detto DOM) è un'interfaccia per diverse piattaforme e indipendente dal linguaggio che tratta un documento XML o HTML come una struttura ad albero in cui ogni nodo è un oggetto che rappresenta una sua parte. Ogni ramo dell'albero termina con un nodo e ogni nodo contiene oggetti. I metodi DOM consentono l'accesso programmatico all'albero; con essi è possibile modificare la struttura, lo stile o il contenuto di un documento.

```
1:  let initial () =
2:    { Cols = ['A' .. 'K']
3:      Rows = [1 .. 15]
4:      Active = None
5:      Cells = Map.empty },
6:    Cmd.Empty
7:
8:  Program.mkProgram initial update view
9:  |> Program.withReact "main"
10: |> Program.run
```

Lo stato iniziale definisce gli intervalli di righe e colonne disponibili e specifica che non ci sono valori in nessuna cella. Quindi usiamo **mkProgram** per comporre tutti gli elementi, specifichiamo **React** come motore di esecuzione e avviamo l'applicazione **Elmish**

facile di quanto ci si possa aspettare! Innanzitutto, vediamo come valutare le formule. All'inizio, abbiamo definito il tipo `Expr` come un'**unione discriminata** con tre casi: **Number**, **Binary** e **Reference**. Per valutare un'espressione, è necessario scrivere una **funzione ricorsiva** che utilizzi il **pattern matching** e affronti in modo appropriato ogni caso. Iniziamo con una versione semplice che non gestisce gli errori e non controlla le formule ricorsive:

```
let rec evaluate cells expr =
  match expr with
  | Number num ->
    num
  | Binary(l, op, r) ->
    let ops = dict [ '+', (+); '-', (-); '*', (*); '/', (/) ]
    let l, r = evaluate cells l, evaluate cells r
    ops.[op] l r
  | Reference pos ->
    let parsed = parse (Map.find pos cells)
    evaluate cells (Option.get parsed)
```

La funzione prende le celle del foglio di calcolo come primo argomento, perché potrebbe dover consultare i valori delle celle a cui fa riferimento l'espressione corrente. Prende anche l'espressione `expr` e la confronta con un **pattern**. Gestire **Number** è facile: basta restituire il numero. La gestione di **Binary** è un po' più interessante, perché dobbiamo richiamare `evaluate` in modo ricorsivo per valutare i valori delle sottoespressioni sinistra e destra. Dopo averli ottenuti, utilizziamo un semplice dizionario per **mappare** l'operatore in una funzione (scritta utilizzando gli operatori standard di F#) ed eseguirla. Infine, quando si gestisce una **Reference**, per prima cosa si ottiene l'input nella cella data, si fa il parsing e poi (di nuovo) si chiama ricorsivamente `evaluate`. Questa operazione può fallire in molti modi: la cella potrebbe essere vuota o il `parser` potrebbe non funzionare. Miglioreremo questo aspetto nella prossima versione del nostro valutatore, in cui la funzione restituisce `int option` anziché `int`. Il valore mancante `None` indica che qualcosa è andato storto.

TIP

Il pattern matching verifica la presenza di una certa struttura (pattern) all'interno di un oggetto composito. Il pattern può essere riconosciuto all'interno di una stringa di caratteri con vari algoritmi, oppure in strutture dati astratte come liste o alberi.

```
let rec evaluate visited cells expr =
  match expr with
  | Number num ->
    Some num

  | Binary(l, op, r) ->
    let ops = dict [ '+', (+); '-', (-); '*', (*); '/', (/) ]
    evaluate visited cells l |> Option.bind (fun l ->
      evaluate visited cells r |> Option.map (fun r ->
        ops.[op] l r ))

  | Reference pos when Set.contains pos visited ->
    None

  | Reference pos ->
    Map.tryFind pos cells |> Option.bind (fun value ->
      parse value |> Option.bind (fun parsed ->
        evaluate (Set.add pos visited) cells parsed))
```

Nel caso di Number, ora viene restituito `Some num` e la valutazione non può fallire. Con Binary, entrambe le chiamate ricorsive possono fallire e si ottengono due valori di `option`. Per gestirlo, utilizziamo `Option.bind` e `Option.map`: entrambi richiamano la funzione specificata solo se l'operazione precedente ha avuto successo, altrimenti restituiscono immediatamente `None`, indicando un errore. Se entrambe le sottoespressioni di sinistra e di destra possono essere valutate, possiamo applicare l'operatore numerico binario ai loro risultati. La gestione dei riferimenti è simile: mettiamo in sequenza una serie di operazioni che possono fallire usando `Option.bind`. Un'altra caratteristica interessante aggiunta in questa versione è il controllo dei riferimenti ricorsivi. A tale scopo, la funzione `evaluate` accetta ora il parametro `visited`, che è un insieme di celle a cui si è acceduto durante la valutazione. Aggiungiamo le celle all'insieme utilizzando `Set.add pos visited` alla **riga 18**. Quando troviamo un riferimento a una cella che abbiamo già visitato (**riga 12**), restituiamo immediatamente `None`, perché ciò porterebbe a un **ciclo infinito**.

Eseguire il parsing delle formule

Infine, l'ultima parte della logica che dobbiamo implementare è il **parsing** delle formule inserite dall'utente in valori del nostro tipo `Expr`. A tale scopo, utilizzeremo una **libreria parser combinator** molto semplice (che potete trovare nel codice sorgente completo). I suoi concetti chiave sono quattro:
- `Parser<char, 'T>` rappresenta un parser che prende come input un elenco di caratteri. Restituisce `None` se il parser non può analizzare l'input. Altrimenti il parser analizza un valore e lo restituisce insieme al resto dell'input. Il fatto che i parser non debbano gestire l'intero input rende facile la loro composizione.
- `<*>` è un operatore binario che prende due parser; esegue il primo (ottenendo un valore di tipo `'T1`) e poi lancia il secondo sul resto dell'input, ottenendo un valore di tipo `'T2`. Ha successo solo

se entrambi i parser danno esito positivo e restituisce una coppia con entrambi i valori.
- **<|>** è un operatore binario che richiede due parser, ma entrambi devono riconoscere valori dello stesso tipo. Prova a eseguire il primo e, se fallisce, tenta di usare il secondo. Ha successo se uno dei due parser ha esito positivo e restituisce ciò che quest'ultimo ha generato.
- **map**, infine, è una funzione che trasforma il valore prodotto da un parser. Dato un parser di tipo **Parser<'T>** e una funzione **'T -> 'R**, restituisce un parser che esegue quello originale e, se ha successo, applica la funzione al risultato.

Lo **snippet** seguente mostra come utilizzare queste idee per creare semplici parser per riconoscere operatori, riferimenti e numeri:

```
1:
2:
3:
let operator = char '+' <|> char '-' <|> char '*' <|> char '/'
let reference = letter <*> integer |> map Reference
let number = integer |> map Number
```

La funzione **char** crea un parser che riconosce solo il carattere dato (e lo restituisce come risultato). Pertanto, il parser dell'operatore riconosce i quattro operatori binari numerici standard e non accetta altri caratteri. Il parser dei riferimenti riconosce una lettera seguita da un numero. Questo restituisce una coppia **char * int** che viene trasformata nel valore di riferimento di **Expr** utilizzando la funzione **map**. Il parsing di un numero è ancora più semplice: basta eseguire il parser integrato per gli interi e avvolgerlo in **Number**. Si noti che il tipo di **Reference** e di **Number** è ora lo stesso: **Parser<char, Expr>**. Ciò significa che possiamo comporli usando **<|>** per creare un parser che riconosca uno dei due tipi di espressione. Il resto del parsing è un po' più complicato, perché dobbiamo gestire le parentesi come **(1+2)*3** e ignorare gli **spazi bianchi**, ma i concetti sono gli stessi:

```
let exprSetter, expr = slot ()
let brack =
  char '(' <*>> anySpace <*>> expr <<*> anySpace
<<*> char ')'
let term = number <|> reference <|> brack
let binary =
  term <<*> anySpace <*> operator <<*> anySpace
<*> term
  |> map (fun ((l,op), r) -> Binary(l, op, r))
```

```
1:  let rec evaluate cells expr =
2:    match expr with
3:    | Number num ->
4:        num
5:
6:    | Binary(l, op, r) ->
7:        let ops = dict [ '+', (+); '-', (-); '*', (*); '/', (/) ]
8:        let l, r = evaluate cells l, evaluate cells r
9:        ops.[op] l r
10:
11:   | Reference pos ->
12:       let parsed = parse (Map.find pos cells)
13:       evaluate cells (Option.get parsed)
```

Per valutare un'espressione è necessario scrivere una funzione ricorsiva che utilizzi il **pattern matching** e gestisca in modo appropriato ogni caso

```
1:  let rec evaluate visited cells expr =
2:    match expr with
3:    | Number num ->
4:        Some num
5:
6:    | Binary(l, op, r) ->
7:        let ops = dict [ '+', (+); '-', (-); '*', (*); '/', (/) ]
8:        evaluate visited cells l |> Option.bind (fun l ->
9:          evaluate visited cells r |> Option.map (fun r ->
10:           ops.[op] l r ))
11:
12:   | Reference pos when Set.contains pos visited ->
13:       None
14:
15:   | Reference pos ->
16:       Map.tryFind pos cells |> Option.bind (fun value ->
17:         parse value |> Option.bind (fun parsed ->
18:           evaluate (Set.add pos visited) cells parsed))
```

Quando si gestisce un riferimento, si ottiene prima l'**input** nella cella data, lo si analizza e poi si richiama ricorsivamente **evaluate**

```
let exprAux = binary <|> term
exprSetter.Set exprAux
```

Per gestire la **ricorsione**, la libreria ci permette di creare un parser tramite **slot**, di usarlo e di definire cosa sia in seguito mediante **exprSetter**. Nel nostro caso, definiamo **expr** alla **riga 1**, lo usiamo quando definiamo **brack** (riga 3) e poi lo definiamo alla riga 9. Questo è un riferimento ricorsivo; **exprAux** può essere binary, che contiene **term**, che può essere **brack** e che, a sua volta, contiene **expr**. L'unica altra cosa ingegnosa dello snippet sono gli operatori **<<*>** e **<*>>**. Questi si comportano come **<*>**, ma restituiscono solo il risultato del parser a sinistra o a destra (dove punta la doppia freccia). Questo è utile, perché possiamo scrivere **anySpace <*>> expr <<*> anySpace** per fare parsing di un'espressione circondata da spazi bianchi, ma otteniamo un parser che restituisce solo il risultato di **expr** (non ci interessa cosa fossero gli spazi bianchi). Infine, definiamo una formula che è **=** seguito da un'espressione e da un'equazione, cioè ciò che si può digitare nel foglio di calcolo, ossia una formula oppure un numero.

```
let formula = char '=' <*>> anySpace <*>> expr
let equation = anySpace <*>> (formula <|> number)
<<*> anySpace
let parse input = run equation input
```

La funzione **parse** definita nell'ultima riga ci permette di eseguire il parser principale delle equazioni su un dato input. Prende una sequenza di caratteri e produce **option<Expr>**, ossia esattamente ciò che abbiamo usato in precedenza nell'articolo. L'applicazione finale del foglio di calcolo è piuttosto semplice ma ha una serie di aspetti interessanti. Dal punto di vista tecnico, ha un'interfaccia utente in cui è possibile selezionare e modificare le celle, analizza le formule inserite e le valuta, gestendo errori e riferimenti ricorsivi. Il codice sorgente completo si trova su **GitHub** **https://github.com/tpetricek/elmish-spreadsheet**/. Il **repository** è concepito come un esercizio pratico in cui è possibile iniziare con un modello, portare a termine una serie di attività e finire con un foglio di calcolo, ma c'è anche un ramo in cui si trova il codice sorgente finito. LXP

TIP

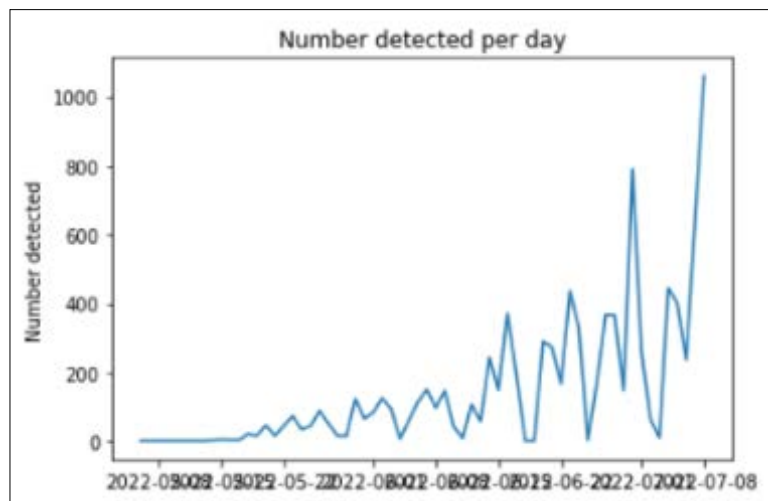
Il parsing, detto anche analisi sintattica o parsificazione, è un processo che analizza un flusso di dati di input per determinare la correttezza della sua struttura grazie a una data grammatica formale. Un parser è un programma che esegue questo compito.

PYTHON

Crediti: Tracyrenee https://medium.com/@tracyrenee61
https://bit.ly/3S54nYK

Serie storiche per analizzare un virus

Come studiare l'impatto globale del virus del vaiolo delle scimmie analizzando dei dataset con una serie di librerie

Secondo i dati del servizio sanitario nazionale degli **Stati Uniti** (**NHS**), il vaiolo delle scimmie è una malattia infettiva rara che si trova principalmente nell'**Africa** occidentale o centrale. Questo virus può essere contratto da roditori infetti, venendone morsi o toccandone il sangue, i fluidi corporei, le macchie, le vesciche o le croste. È possibile prenderlo anche mangiando la carne di un animale infetto o toccandolo. Il virus può essere diffuso tra esseri umani entrando in contatto con gli indumenti, la biancheria da letto o gli asciugamani di una persona infetta, toccando la pelle, le vesciche o le croste di un malato o trovandosi a breve distanza da una persona contagiata che tossisce o starnutisce. Poiché ho scritto una serie di articoli sul **COVID19**, questa estate ho voluto esaminare anche i dati sull'impatto globale del vaiolo delle scimmie. Per farlo ho utilizzato il **dataset** che si trova sul sito di **Kaggle** all'indirizzo **https://bit.ly/3CBapL1**.

Un grafico realizzato con la libreria **matplotlib** (**https://matplotlib.org/**) per creare visualizzazioni in **Python**. Raggruppa le infezioni in base alla data in cui sono state registrate

Gli strumenti dell'analisi

Ho aperto un **Jupyter Notebook** sul sito di **Kaggle** e ho importato le librerie di cui avrei avuto bisogno. Si tratta di **NumPy** per eseguire calcoli numerici e creare **array numpy**, **Pandas** per creare e manipolare **dataframe**, **Matplotlib** per tracciare i punti di dati sui grafici, **Seaborn** per gestirli statisticamente e il modulo **os** per recuperare i file **CSV** dal sistema di Kaggle:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file 1/0
(e.g. pd. read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
```

Una volta importate le librerie, ho usato **os** per recuperare i tre **file CSV** usati nel programma, ossia **Monkey_Pox_Cases_Worldwide.csv**, **Worldwide_Case_Detection_Timeline.csv** e **Daily_Country_Wise Confirmed_Cases.csv**, presi da Kaggle.

```
import os
        for dirname, -, filenames in os.walk('
kaggle/input')
                for filename in filenames:
                        print(os.path
join(dirname, filename))
```

Dopo aver letto i file CSV nel programma, ho usato Pandas per convertire i file in dataframe:

```
worldwide = pd. read_csv("/kaggle/input
monkeypox-dataset-daily-updated/Monkey_Pox
Cases_Worldwide.csv") worldwide.head(10)
```

Volevo vedere quali Paesi hanno inviato i dati sulle infezioni da vaiolo delle scimmie, quindi ho fatto l'elenco di quelli presenti nella colonna **Country** del dataframe **detection_timeline**. Ho quindi creato un set vuoto (chiamato **set_country**) e un **ciclo for** che **iterava** l'elenco e metteva tutti i nomi degli stati in un set. Quest'ultimo contiene solo un'occorrenza di ciascun nome di Paese.

```
country = detection_timeline[ 'Country' ].to_list()
```

```
set_country = set() for i in country:
           set_country.add(i)

```

Una volta ottenuto il set che conteneva il nome di ciascuna nazione rappresentata nel dataframe, ho controllato la sua lunghezza, concludendo che **61** Paesi stavano inviando dati sul vaiolo delle scimmie:

```
len(set_country)
61
```

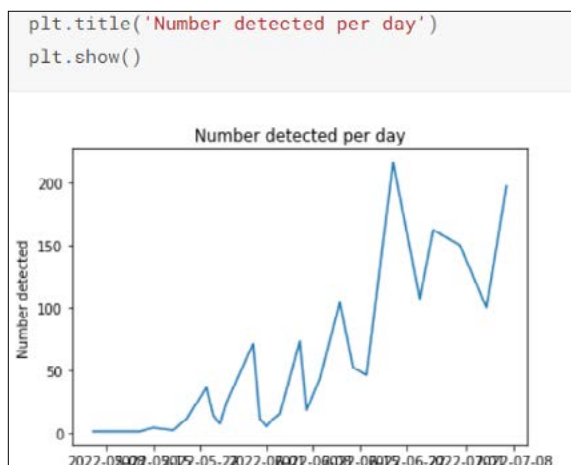
Le aree con maggior incidenza

Ho deciso di creare una colonna aggiuntiva, **num_detected**, all'interno del dataframe **detection_timeline**. Ogni cella di questa colonna conteneva il valore **1**, per riflettere ogni volta che veniva registrata un'istanza. Ho quindi utilizzato la funzione **groupby** per creare la variabile **infection_per_country**. Una volta raggruppata ogni infezione in base al Paese in cui è stata identificata, le ho ordinate in ordine discendente per mostrare le incidenze delle infezioni dalla maggiore alla minore. I tre maggiori focolai sono stati **Spagna**, **Germania** e **Inghilterra**.

```
detection_timeline['num_detected'] = 1

infection_per_country = detection_timeline
groupby(['Country']).sum() pd.set_option('display
max_rows', None)
infection-per_country.sort_values (by=['num
detected'], inplace=True, ascending=False)
infection_per_country
```

Ho quindi raggruppato le infezioni in base alla data in cui sono state registrate. Ho definito la variabile in cui memorizzare questi dati con il nome **infection_per_day**:

```
infection_per_day = detection_timeline
groupby(['Date_confirmation']). sum() pd.set
option('display.max_rows', None)
infection_per_day
```

Ho poi utilizzato matplotlib per raggruppare le infezioni in un **grafico.** Dall'immagine della prima pagina di questo articolo si può notare che il numero di infezioni tendeva ad aumentare:

Dopo aver filtrato l'Inghilterra dal **dataframe** sono stati creati una **tabella pivot** e un grafico con i **punti dati**

» SCIENZA DEI DATI PER TUTTI

Kaggle (**https://www.kaggle.com/**) è una comunità online dedicata alla **scienza dei dati** e all'**apprendimento automatico**. Permette agli utenti di trovare e pubblicare set di dati, esplorare e costruire modelli, collaborare con altri studiosi e programmatori e partecipare a concorsi per risolvere sfide di scienza dei dati. Offre un ambiente **Jupyter Notebooks** che non richiede configurazione iniziale ma è personalizzabile e consente l'accesso alle **GPU** senza alcun costo. La quantità di dati e codice pubblicati dalla comunità è molto vasta. Ci trovate **dataset** su svariati argomenti, dalla finanza ai temi sociali. Kaggle ha iniziato la sua attività nel **2010** proponendo gare di apprendimento automatico e ora offre anche una piattaforma di dati pubblica, un **workbench** basato sul **cloud** per la scienza dei dati e formazione sull'intelligenza artificiale. È controllato da **Google LLC**.

```
plt.plot(infection_per_day.index, infection_per_day
num_detected)
plt.xlabel('Data')
plt.ylabel('Numero rilevato')
plt.title('Numero rilevato al giorno')
plt.show()
```

Poiché vivo nel **Regno Unito**, ho pensato di esaminare l'incidenza delle infezioni da vaiolo delle scimmie in quel Paese. Ho filtrato il Regno Unito dal dataframe **detection_timeline**, chiamando la nuova variabile creata in questo processo **uk_pox**:

```
country = "Regno Unito"

include_uk = detection_timeline [detection_timelines
'Country').values == country] exclude_uk =
detection_timeline detection_timeline Country')
values != country]
uk-pox = include_uk
uk-pox.head (10)
```

Ho usato quindi Pandas per creare una **tabella pivot** sul numero di incidenze del virus rilevate alla data della conferma, chiamandola **uk_pivot**:

```
uk_table = pd.pivot_table(uk_pox, values='num
detected', index=( 'Date_confirmation'), aggfunc=np.
sum) uk_table
```

A questo punto ho creato un grafico con i **punti dati** nella tabella **uk_table** e nell'illustrazione in questa pagina si può notare che l'incidenza delle infezioni era in aumento, con il massimo numero di registrazioni al **20 giugno 2022** (**216**):

```
plt.plot(uk_table.index, uk_table.num_detected)
plt.xlabel('data')
plt.ylabel( 'Numero rilevato')
plt.title('Numero rilevato al giorno')
plt.show()
```

Ho anche preparato una revisione video del codice del dataset del vaiolo delle scimmie, che può essere consultata su **YouTube** all'indirizzo **https://bit.ly/3yL3jSV**. Buono studio dei dati da tutti!

TIP

Una tabella pivot aggrega i singoli elementi di una tabella più ampia (come quella di un database o di un foglio elettronico) all'interno di una o più categorie discrete.

PROGRAMMAZIONE

Crediti: Monday Morning Haskell
https://mmhaskell.com/

Definire i propri tipi di dati in Haskell

Capire il sistema dei tipi è una parte molto importante dell'apprendimento di questo linguaggio e vi permette di ampliare le vostre possibilità

Comprendere e definire i diversi tipi di dati in **Haskell** è la base per poter sfruttare appieno questo affascinante linguaggio di programmazione puramente **funzionale**.
Per questo articolo, supponiamo di voler modellare l'elenco delle cose da fare di un utente. Risulta quindi necessario creare diversi tipi di dati `Task` per rappresentare ogni singolo compito della lista.
Per creare un tipo di dati si usa innanzitutto la parola chiave `data`, seguita dal nome del tipo. Poi si aggiunge l'**operatore di assegnazione** `=`:

```
module DataTypes where

data Task1 = ...
```

Si noti che, a differenza delle espressioni e dei nomi delle funzioni, il tipo inizia con la lettera maiuscola.
Ora creeremo il nostro primo **costruttore**, ossia un tipo speciale di espressione che vi consente di creare il vostro oggetto del tipo `Task` e contiene un elenco di tipi. In questo caso, vogliamo che il nostro task abbia un nome e una durata prevista (in minuti). Rappresenteremo il nome con una stringa e la durata del tempo con un `Int`.

```
data Task1 = BasicTask1 String Int
```

Così si può iniziare a creare oggetti `Task`!
Per esempio, definiamo un paio di task di base come espressioni all'interno del modulo:

```
assignment1 :: Task1
assignment1 = BasicTask1 "Svolgi il compito 1" 60

laundry1 :: Task1
0 "Fai il bucato" 45
```

Si può anche caricare il codice nell'**interprete** per verificare che si compili ancora e abbia senso:

```
>> :l MyData.hs
>> :t assignment1
assignment1 :: Task1
>> :t laundry1
laundry1 :: Task
```

TIP

In Haskell il tipo di ogni espressione è noto in fase di compilazione, il che porta a un codice più sicuro. Se si scrive un programma in cui si cerca di dividere un tipo booleano per un numero, non verrà nemmeno compilato. In questo modo è possibile individuare gli errori in fase di compilazione, evitando il rischio di far crashare il programma.

```
1  module DataTypes where
2
3  import Data.Char (toUpper)
4
5  data Location =
6    School |
7    Office |
8    Home
9
10 schoolLocation :: Location
11 schoolLocation = School
12
13 officeLocation :: Location
14 officeLocation = Office
15
16 homeLocation :: Location
17 homeLocation = Home
```

Definiamo un tipo per i vari luoghi in cui si può svolgere un compito, creando un **costruttore** per ciascuno di essi e separandoli con la barra verticale **|**

Si noti che il tipo dell'espressione è `Task1`, anche se gli oggetti utilizzano il costruttore `BasicTask1`. In **Java**, si possono avere molti costruttori per lo stesso tipo. Si può fare anche in Haskell, ma l'aspetto è un po' diverso. Definiamo un altro tipo per i diversi luoghi in cui si può svolgere un compito, per esempio a scuola, in ufficio o a casa. Lo rappresenteremo creando un costruttore per ciascuno di essi. Separiamo i costruttori utilizzando la barra verticale **|**:

```
data Location =
  School |
  Office |
  Home
```

In questo caso, ogni costruttore è un semplice **marcatore** che non ha parametri o dati memorizzati al suo interno. Questo è un esempio di tipo `Enum`.
Tecnicamente, si possono creare diversi tipi di espressioni che rappresentino ciascuno di essi:

```
schoolLocation :: Location
schoolLocation = School

officeLocation :: Location
```

```
officeLocation = Office

homeLocation :: Location
homeLocation = Home
```
Queste espressioni, però, non sono più utili dell'uso dei costruttori stessi. Ora che ci sono due tipi diversi, si può fare in modo che uno dei due tipi contenga l'altro! Aggiungeremo un nuovo costruttore al nostro tipo Task. Rappresenterà un task più complicato che indica anche una posizione:
```
data Task1 =
  BasicTask1 String Int |
  ComplexTask1 String Int Location
...

complexTask :: Task1
complexTask = ComplexTask1 "Scrivi memo" 30 Office
```
Questo si stacca molto dai costruttori di altri linguaggi. Si possono avere campi differenti per rappresentazioni diverse del tipo, includendo dati completamente differenti a seconda del costruttore utilizzato. In questo modo si ottiene una grande flessibilità, che altri linguaggi faticano a concedere.

Usare i tipi parametrizzati

Un'altra possibilità che si può avere con le definizioni dei tipi è quella di usare i parametri dei tipi.
Ciò significa che uno o più campi dipendono da un tipo che il programmatore può selezionare. Supponiamo di avere un tipo che ha alcuni costruttori di base per diverse quantità di tempo. Questo limiterebbe la descrizione del tempo per motivi di semplicità.
```
data TaskLength =
  QuarterHour |
  HalfHour |
  ThreeQuarterHour |
  Hour |
  HourAndHalf |
  TwoHours |
  ThreeHours
```
Ora potremmo voler descrivere un'attività in cui la lunghezza dell'attività è un `Int` o anche volere che il task sia in grado di usare questo nuovo tipo di lunghezza dell'attività. Creiamo una seconda versione del nostro tipo `Task`, che possa usare entrambi i tipi per la lunghezza. Si può fare **parametrizzando** il tipo in questo modo:
```
data Task2 a =
  BasicTask2 String a |
  ComplexTask2 String a Location
```
Il tipo `a` è ora un tipo ignoto, che possiamo riempire a piacimento. Ora, però, ogni volta che elenchiamo il tipo `Task2` in una firma di tipo, dobbiamo inserire la definizione appropriata:
```
assignment2 :: Task2 Int
assignment2 = BasicTask2 "Svolgi il compito 2" 60

assignment2' :: Task2 TaskLength
assignment2' = BasicTask2 "Svolgi il compito 2" Hour

laundry2 :: Task2 Int
laundry2 = BasicTask2 "Fai il bucato" 45

laundry2' :: Task2 TaskLength
laundry2' = BasicTask "Fai il bucato" ThreeQuarterHour

complexTask2 :: Task2 TaskLength
complexTask2 = ComplexTask2 "Scrivi memo"
HalfHour Office
```
Bisogna d'altro canto fare attenzione, perché questo può limitare le possibilità di eseguire alcune operazioni. Per esempio, non si può creare un elenco che contenga sia `assignment2` sia `complexTask2`, perché le due espressioni hanno ora tipi diversi:
```
-- QUESTO CAUSERÀ UN ERRORE DEL COMPILATORE
badTaskList :: [Task2 a]
badTaskList = [assignment2, complexTask2]
```

Implementare una lista

Ci sono variazioni sintattiche nel modo in cui scriviamo le liste nella pratica ma, a livello di sorgente, sono definite da due costruttori: `Nil` e `Cons`.
```
>data List a =
  Nil |
  Cons a (List a)
```
Come prevedibile, il tipo `List` ha un singolo parametro di tipo. Il costruttore `Nil` è un elenco vuoto che non contiene oggetti. Quindi, ogni volta che si utilizza l'espressione `[]`, si sta utilizzando `Nil`. Il secondo costruttore concatena un singolo elemento con un'altra

```
data Task1 =
  BasicTask1 String Int |
  ComplexTask1 String Int Location
...

complexTask :: Task1
complexTask = ComplexTask1 "Write Memo" 30 Office
```
Un nuovo **costruttore** che rappresenta un **task** più complicato che indica anche una posizione

» UN SITO PER SCOPRIRE TUTTI I SEGRETI DI HASKELL

Questo **tutorial** sulla creazione dei tipi in **Haskell** è tratto dal materiale del sito **https://mmhaskell.com/liftoff**, che offre una serie di guide gratuite che spiegano vari temi, da come iniziare a programmare nel linguaggio a come usare le **API** e comprendere le **strutture dati**. La serie è accompagnata da un **repository GitHub** che vi permetterà di lavorare con alcuni esempi di codice dei suoi articoli. Trovate quello per il materiale usato in queste pagine agli indirizzi **https://bit.ly/3EqCx5m** e **https://bit.ly/3SG3GpH**. Sul sito **Monday Morning Haskell!**, oltre a numerosi tutorial gratuiti per ogni livello, sono proposti anche vari corsi a pagamento. C'è inoltre un **blog** ricco di spunti interessanti e, iscrivendovi con la vostra **email,** otterrete l'accesso a una serie di risorse scaricabili. Il sito è in inglese ma vale una visita!

TIP

Un tipo di dato astratto o ADT (Abstract Data Type) è un modello matematico per i tipi di dati definito dal suo comportamento dal punto di vista di un utente, in particolare in termini di valori e operazioni possibili e comportamento di queste ultime. Questo modello matematico si contrappone alle strutture dati, che sono rappresentazioni concrete dei dati e rappresentano il punto di vista di un implementatore.

lista. Il tipo dell'elemento e dell'elenco devono ovviamente corrispondere. Quando si usa l'operatore `:` per aggiungere un elemento all'inizio di un elenco, si impiega in realtà il costruttore `Cons`.

```
emptyList :: [Int]
emptyList = [] -- Actually Nil

fullList :: [Int]
-- Equivalent to Cons 1 (Cons 2 (Cons 3 Nil))
-- More commonly written as [1,2,3]
fullList = 1 : 2 : 3 : []
```

Un altro aspetto interessante è che la nostra struttura di dati è ricorsiva. Si può vedere nel costruttore `Cons` come una lista contenga un'altra lista come parametro. Questo funziona bene, purché ci sia un caso base. In questa situazione, abbiamo `Nil`. Immaginate se avessimo un solo costruttore che accetta un parametro ricorsivo. Sarebbe complesso creare una qualsiasi lista!

Sfruttare la sintassi di record

Torniamo al nostro tipo di dati `Task` di base, non parametrizzato. Supponiamo che non ci interessi l'intero elemento del task, ma piuttosto uno dei suoi componenti, come il nome o l'ora. Allo stato attuale del codice, l'unico modo per farlo è utilizzare il **pattern matching** per rivelare questi campi.

```
import Data.Char (toUpper)

...

twiceLength :: Task1 -> Int
twiceLength (BasicTask1 name time) = 2 * time

capitalizedName :: Task1 -> String
capitalizedName (BasicTask1 name time) = map toUpper
name

tripleTaskLength :: Task1 -> Task1
tripleTaskLength (BasicTask1 name time) = BasicTask1
name (3 * time)
```

Ora possiamo semplificare un po' il tutto. Si possono utilizzare i trattini bassi al posto dei parametri che non si usano. Ma anche in questo caso, può diventare molto complicato se si ha un tipo di dati con molti campi. Potremmo scrivere delle funzioni che ci permettano di accedere ai singoli campi. Anche queste funzioni dovranno utilizzare il pattern matching:

```
taskName :: Task1 -> String
taskName (BasicTask1 name _) = name

taskLength :: Task1 -> Int
```

```
import Data.Char (toUpper)

...

twiceLength :: Task1 -> Int
twiceLength (BasicTask1 name time) = 2 * time

capitalizedName :: Task1 -> String
capitalizedName (BasicTask1 name time) = map toUpper name

tripleTaskLength :: Task1 -> Task1
tripleTaskLength (BasicTask1 name time) = BasicTask1 name (3 * time)
```

Se siete interessati a un componente del task, come il nome o l'ora, potete utilizzare il **pattern matching** per rivelarne i campi

```
emptyList :: [Int]
emptyList = [] -- Actually Nil

fullList :: [Int]
-- Equivalent to Cons 1 (Cons 2 (Cons 3 Nil))
-- More commonly written as [1,2,3]
fullList = 1 : 2 : 3 : []
```

Quando si usa l'operatore **:** per aggiungere un elemento all'inizio di un elenco, si impiega in realtà il costruttore **Cons**

```
taskLength (BasicTask1 _ time) = time

twiceLength :: Task1 -> Int
twiceLength task = 2 * (taskLength task)

capitalizedName :: Task1 -> String
capitalizedName task = map toUpper (taskName task)

tripleTaskLength :: Task1 -> Task1
tripleTaskLength task = BasicTask1 (taskName task) (3 *
(taskLength task))
```

Questo approccio, però, non è scalabile, poiché dovremmo scrivere queste funzioni per ogni campo diverso di ogni tipo di dato che creiamo. Possiamo però fare in modo che Haskell scriva queste funzioni al posto nostro, utilizzando la sintassi dei record. Per farlo, dobbiamo solo assegnare a ogni campo un nome nella nostra definizione dei dati. Creiamo una nuova versione di `Task`:

```
data Task3 = BasicTask3
  { taskName :: String
  , taskLength :: Int }
```

Ora possiamo scrivere lo stesso codice senza le **funzioni getter** che abbiamo scritto sopra.

```
-- Questi funzioneranno ora SENZA le definizioni separate
per "taskName" e "taskName"
-- "taskLength"
twiceLength :: Task3 -> Int
twiceLength task = 2 * (taskLength task)

capitalizedName :: Task3 -> String
capitalizedName task = map toUpper (taskName task)
```

Ora, quando costruiamo i task, possiamo ancora usare il costruttore `BasicTask3` da solo. Per maggiore chiarezza del codice, possiamo anche inizializzare l'oggetto usando la sintassi dei record, in cui si assegna un nome al campo:

```
-- Anche asicTask3 "Svolgi il compito 3" 60 andrebbe
bene
assignment3 :: Task3
assignment3 = BasicTask3
  { taskName = "Svolgi il compito 3"
  , taskLength = 60 }

laundry3 :: Task3
laundry3 = BasicTask3
  { taskName = "Fai il bucato"
  , taskLength = 45 }
```

Possiamo anche scrivere una **funzione setter** in modo più semplice, utilizzando la sintassi dei

record. Utilizziamo il task precedente e poi un elenco di "modifiche" da apportare all'interno delle **parentesi graffe**:

```
tripleTaskLength :: Task3 -> Task3
tripleTaskLength task = task { taskLength = 3 *
(taskLength task) }
```

In genere, si usa la sintassi dei record solo quando c'è un unico costruttore per un tipo di dati. Possiamo usare campi diversi per costruttori diversi, ma il codice diventa un po' meno sicuro. Vediamo un altro esempio di definizione di `Task`:

```
data Task4 =
  BasicTask4
    { taskName4 :: String,
      taskLength4 :: Int }
  |
  ComplexTask4
    { taskName4 :: String,
      taskLength4 :: Int,
      taskLocation4 :: Location }
```

Il problema di questo sistema è che il compilatore genererà una funzione `taskLocation4` che verrà compilata per qualsiasi task, ma sarà valida solo se richiamata per un `ComplexTask4`. Quindi il codice seguente verrà compilato ma poi causerà un arresto anomalo:

```
>causeError :: Location
causeError = taskLocation4 (BasicTask4 "Causa errore" 10)
```

Inoltre, se i nostri diversi costruttori usano tipi differenti, non possiamo assegnare loro lo stesso nome, il che può essere frustrante quando si vuole rappresentare lo stesso concetto con tipi diversi. Questo esempio non verrà compilato perché **GHC** (il compilatore Haskell più usato) non è in grado di determinare il tipo della funzione `taskLength4`. Potrebbe infatti essere di tipo **Task -> Int** o **Task -> TaskLength**.

```
data Task4 =
  BasicTask4
    { taskName4 :: String,
      taskLength4 :: Int }
  |
  ComplexTask4
    { taskName4 :: String,
      taskLength4 :: TaskLength, -- Si noti che qui si usa
"TaskLength" e non un Int!
      taskLocation4 :: Location }
```

La parola chiave Type

Ora conoscete la maggior parte dei dettagli sulla creazione dei propri tipi di dati, ma ci sono casi in cui non è necessario farlo. Si possono generare nuovi nomi di tipi senza creare una struttura di dati completamente nuova. Un metodo per farlo è la parola chiave `type`. Essa consente di creare un **sinonimo di tipo**, come la parola chiave `typedef` in **C++**. Il più comune, come abbiamo visto, è che una stringa è in realtà un elenco di caratteri:

```
type String = [Char]
```

Un caso d'uso comune è quando si combinano molti tipi diversi in una **tupla**. Può essere piuttosto noioso scriverla più volte nel codice:

```
makeTupleBigger :: (Int, String, Task) -> (Int, String,
Task)
makeTupleBigger (intValue, stringValue, (BasicTask
name time) =
  (2 * intValue, map toUpper stringValue, (BasicTask (map
toUpper name) (2 * time)))
```

Un **sinonimo di tipo** renderebbe di fatto il codice molto più pulito:

```
type TaskTuple = (Int, String, Task)

makeTupleBigger :: TaskTuple -> TaskTuple
makeTupleBigger (intValue, stringValue, (BasicTask
name length) =
  (2 * intValue, map toUpper stringValue, (BasicTask (map
toUpper name) (2 * length))
```

Naturalmente, se questo insieme di elementi si presenta spesso, potrebbe valere la pena di creare un tipo di dati completo. Inoltre, i sinonimi di tipo non sono sempre la scelta migliore. Innanzitutto, possono portare a errori di compilazione a volte difficili da risolvere. Probabilmente vi sarete già imbattuti in qualche errore in cui il compilatore vi diceva che si aspettava un `[Char]`. Sarebbe stato molto più chiaro se avesse detto **String**. Possono anche portare a del codice poco intuitivo. Supponiamo di usare una tupla di base invece di un tipo di dati per rappresentare un task. Qualcuno potrebbe aspettarsi che il tipo `Task` sia un tipo di dati a sé stante quindi sarà un po' confuso quando lo si manipolerà come una tupla.

```
type Task5 = (String, Int)

twiceTaskLength :: Task5 -> Int
-- "snd task" confonde qui
twiceTaskLength task = 2 * (snd task)
```

Questo conclude la nostra discussione sulla creazione di tipi di dati personalizzati. Per ulteriori informazioni, potete anche visitare **wiki.haskell.org**. LXP

» ALLA SCOPERTA DEI NEWTYPE

I **newtype** sono come i sinonimi di tipo per certi versi e i **tipi di dati astratti** (**ADT**) per altri. Hanno un ruolo importante in Haskell ed è bene conoscerli. Supponiamo di voler avere un nuovo approccio alla rappresentazione di `TaskLength`. Vogliamo usare un numero standard, ma che abbia un suo tipo separato. Possiamo farlo usando `newtype`

```
newtype TaskLength2 = TaskLength2 Int
```

La sintassi dei newtype assomiglia molto alla definizione di un ADT. Tuttavia, la definizione di un newtype può avere solo un unico costruttore, che a sua volta può accettare un singolo parametro di tipo. La grande differenza tra un ADT e un newtype viene dopo la compilazione del codice. In questo esempio, non ci sarà alcuna differenza tra i tipi `TaskLength` e `Int` in fase di esecuzione. Questo è un bene, perché molto codice per i tipi `Int` è specializzato per essere eseguito velocemente. Se si trattasse di un vero ADT, non sarebbe così:

```
data TaskLength2 = TaskLength2 Int
```

Per il resto, con il newtype si possono fare molti degli stessi trucchi che si possono usare con gli ADT. Per esempio, possiamo usare la sintassi dei record nel costruttore del nostro newtype.

TIP

In vari linguaggi di programmazione funzionale, come l'Haskell, Miranda, od OCaml, ecc., è possibile definire i cosiddetti sinonimi di tipo, che corrispondono alle typedef in C. Lo scopo è quello di assegnare dei nomi alternativi a dei tipi di dato esistenti, di solito perché la dichiarazione standard è troppo ingombrante oppure per rendere il codice più facile da riutilizzare.

LUA/LÖVE

Crediti: a327ex
github.com/a327ex/blog/issues/30

Il game loop di un gioco in LUA

Gettate le basi di uno sparatutto con un linguaggio di scripting potente e leggero e un framework mirato a sfruttarlo per creare giochi 2D

BYTEPATH (**https://store.steampowered.com/app/760330/BYTEPATH/**) è uno **sparatutto arcade** rigiocabile che si concentra sulla creazione di **build** basate sul **theorycrafting**. Utilizzate un enorme albero di abilità oltre a numerose classi e navi per creare le vostre **build** e sconfiggere un numero sempre crescente di nemici. L'autore lo descrive come un mix di **Bit Blaster XL** e **Path of Exile**, creato con l'intenzione di espandere il **gameplay** rilassante e coinvolgente del primo con la profondità, la varietà di costruzione e gli elementi da **gioco di ruolo** del secondo. Il videogame è stato creato con **Lua** e **LÖVE** e in questo tutorial vedremo il suo **loop di gioco** spiegato dal suo autore.

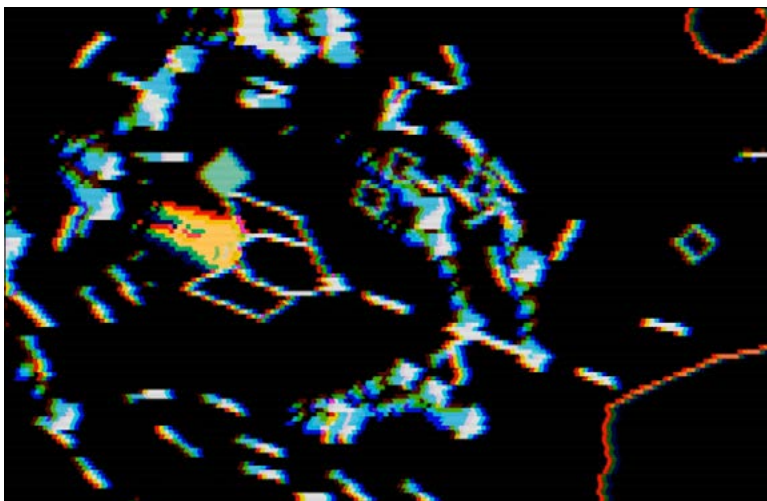
Primi passi con LÖVE

Per iniziare dovete installare **LÖVE**, un **framework** Open Source per creare giochi **2D** in **Lua**, sul vostro sistema da **https://love2d.org**. Potete seguire i passaggi su **https://love2d.org/wiki/Getting_Started** per ulteriori dettagli. Dopo averlo fatto, create un file **main.lua** nella cartella del progetto con i seguenti contenuti:

```
function love.load()

end

function love.update(dt)

end

function love.draw()

end
```

Eseguendolo, dovrebbe apparire una finestra con una schermata nera. Nel codice sopra riportato, una volta eseguito il progetto LÖVE, la funzione `love.load` viene lanciata una volta all'inizio del programma mentre `love.update` e `love.draw` sono usate a ogni fotogramma. Quindi, per esempio, se volete caricare un'immagine e disegnarla, dovrete fare qualcosa di simile a quanto segue:

```
function love.load()
    image = love.graphics.newImage('image.png')
end

function love.update(dt)

end

function love.draw()
    love.graphics.draw(image, 0, 0)
end
```

`love.graphics.newImage` carica la **texture** dell'immagine nella variabile `image` e poi ogni fotogramma viene disegnato nella posizione **0, 0**. Per vedere che `love.draw` disegna effettivamente l'immagine in ogni fotogramma, provate:

Una foto **BYTEPATH**, lo sparatutto in stile **arcade** sviluppato con **Lua** e **LÖVE** di cui vengono esaminati alcuni aspetti nell'ambito di questo articolo

```
love.graphics.draw(image, love.math.random(0, 800),
love.math.random(0, 600))
```

La dimensione predefinita della finestra è **800 x 600**, quindi l'effetto è di disegnare casualmente l'immagine sullo schermo in modo molto veloce. Si noti che tra un fotogramma e l'altro lo schermo viene ripulito, altrimenti l'immagine che si sta disegnando in modo casuale riempirebbe lentamente l'intero spazio. Questo accade perché LÖVE fornisce un ciclo di gioco predefinito per i suoi progetti che cancella lo schermo alla fine di ogni fotogramma. Esamineremo ora questo **loop** di gioco e il modo in cui è possibile modificarlo.

Come funziona il game loop

Il ciclo di gioco predefinito che **LÖVE** utilizza si trova nella pagina **love.run** (**https://love2d.org/wiki/love.run**) e si presenta come segue:

```
function love.run()
  if love.math then
      love.math.setRandomSeed(os.time())
  end

  if love.load then love.load(arg) end

  -- Non vogliamo che il dt del primo fotogramma
include il tempo impiegato da love.load.
  if love.timer then love.timer.step() end

  local dt = 0

  -- Main loop.
  while true do
    -- Elaborazione eventi.
    if love.event then
        love.event.pump()
        for name, a,b,c,d,e,f in love.event.poll() do
          if name == "quit" then
            if not love.quit or not love.quit() then
              return a
            end
          end
            love.handlers[name](a,b,c,d,e,f)
        end
    end

    -- Aggiornare dt, poiché lo si passerà ad update
      if love.timer then
        love.timer.step()
        dt = love.timer.getDelta()
      end

      -- Richiamare update e disegnare
      if love.update then love.update(dt) end -- will
pass 0 if love.timer is disabled

      if love.graphics and love.graphics.isActive()
then
        love.graphics.clear(love.graphics.
getBackgroundColor())
        love.graphics.origin()
      if love.draw then love.draw() end
```

```
-- Main loop time.
while true do
    -- Process events.
    if love.event then
        love.event.pump()
        for name, a,b,c,d,e,f in love.event.poll() do
            if name == "quit" then
                if not love.quit or not love.quit() then
                    return a
                end
            end
            love.handlers[name](a,b,c,d,e,f)
        end
    end
end
```

Qui inizia il ciclo principale. La prima cosa che viene fatta in ogni fotogramma è l'elaborazione degli eventi

```
        love.graphics.present()
      end

      if love.timer then love.timer.sleep(0.001) end
  end
end)
```

Quando il programma si avvia, viene eseguito `love.run` e da lì parte tutto. Il ciclo è abbastanza ben commentato e si può scoprire cosa fa ogni funzione sulla **wiki** di LÖVE. Ne esamineremo però qui le nozioni di base:

```
if love.math then
  love.math.setRandomSeed(os.time())
end
```

Nella prima riga controlliamo se `love.math` non è `nil`. In Lua tutti i valori sono `true`, tranne `false` e `nil`, quindi la condizione `if love.math` sarà `true` se `love.math` è definito in qualsiasi modo. Nel caso di LÖVE, queste variabili sono impostate per essere abilitate o meno nel file **conf.lua**. Non c'è bisogno di preoccuparsi di questo file per ora ma viene menzionato perché è lì che si possono abilitare o disabilitare singoli sistemi come `love.math`. Per questo c'è un controllo per vedere se è attivato o meno prima di fare qualsiasi cosa con una delle sue funzioni. In generale, se una variabile non è definita in Lua e vi si fa riferimento in qualsiasi modo, restituirà un valore nullo. Quindi, se si chiede `if random_variable`, si otterrà `false`, a meno che non sia stata definita in precedenza come `random_variable = 1`. In ogni caso, se il modulo `love.math` è abilitato (e lo è per impostazione predefinita), il suo **seme** o **seed** viene impostato in base all'ora corrente. Si veda `love.math.setRandomSeed` e `os.time`. Dopo che è stato fatto, viene richiamata la funzione `love.load`:

```
if love.load then love.load(arg) end
```

Con `arg` sono indicati gli argomenti della riga di comando passati all'eseguibile LÖVE quando lancia il progetto. Come si può notare, il motivo per cui `love.load` viene eseguito una sola volta è che viene richiamato una sola volta, mentre le funzioni `update` e `draw` vengono lanciate più volte all'interno di un ciclo (e ogni **iterazione** del **loop** corrisponde a un fotogramma).

```
-- Non vogliamo che il dt del primo fotogramma
include il tempo impiegato da love.load.
if love.timer then love.timer.step() end

local dt = 0
```

TIP

Nella wiki di LÖVE (https://bit.ly/3WLYP92), trovate molte indicazioni, guide e tutorial per sfruttare al meglio questo framework. All'indirizzo https://love2d.org/forums/ ci sono anche dei forum per farvi aiutare dalla community se incontrate delle difficoltà.

TIP

Se vi serve un ripasso delle basi di Lua o una breve introduzione pratica al suo funzionamento, potete fare riferimento al tutorial all'indirizzo https://bit.ly/3UL2rXb che ne spiega i fondamenti in modo schematico ed efficace.

Dopo aver richiamato **love.load** e dopo che questa funzione ha svolto tutto il suo lavoro, passiamo a verificare che **love.timer** sia definito e lanciamo **love.timer.step**, che misura il tempo trascorso tra gli ultimi due fotogrammi. **Love.load** potrebbe richiedere molto tempo per l'elaborazione (perché potrebbe caricare diversi elementi, come immagini e suoni) e questo tempo non dovrebbe essere la prima cosa restituita dalla funzione **love.timer.getDelta** nel primo fotogramma del gioco. Qui **dt** è anche inizializzato a **0**. Le variabili in Lua sono **globali** per impostazione predefinita, quindi con **local dt** si definisce solo l'ambito locale del blocco corrente, che in questo caso è la funzione **love.run**. Per saperne di più sui blocchi, potete andare all'indirizzo **https://www.lua.org/pil/4.2.html**:

```
-- Main loop.
while true do
  -- Elaborazione eventi.
  if love.event then
    love.event.pump()
    for name, a,b,c,d,e,f in love.event.poll() do
      if name == "quit" then
        if not love.quit or not love.quit() then
          return a
        end
      end
      love.handlers[name](a,b,c,d,e,f)
    end
  end
end
```

Qui inizia il ciclo principale. La prima cosa che viene fatta in ogni fotogramma è l'elaborazione degli eventi. **love.event.pump** invia gli eventi alla loro coda. Secondo la sua descrizione, questi eventi sono generati dall'utente, per esempio premendo i tasti, facendo click con il mouse o ridimensionando una finestra. Il ciclo che utilizza **love.event.poll** passa in rassegna la coda degli eventi e gestisce ognuno di essi. **love.handlers** è una tabella di funzioni che richiama i **callback** pertinenti. Per esempio, **love.handlers.quit** invocherà la funzione **love.quit**, se esiste. Una delle caratteristiche di LÖVE è la possibilità di definire nel file **main.lua**

Potete trovare un elenco delle librerie disponibili per **Lua** nella **wiki** di **lua-users.org** all'indirizzo **https://bit.ly/3DQ1tC6**

```
-- in objects/Test.lua
Test = Object:extend()

function Test:new()

end

function Test:update(dt)

end

function Test:draw()

end
```

Un esempio di come si presentano la creazione di una **classe Test** e la sua singola **istanziazione**

dei callback che vengono richiamati quando si verifica un evento. Un elenco completo di tutti i callback è disponibile all'indirizzo **https://love2d.org/wiki/love**. Gli argomenti **a, b, c, d, e, f** passati a **love.handlers[name]** sono tutti i possibili argomenti che possono essere utilizzati dalle funzioni pertinenti. Per esempio, **love.keypressed** accetta come argomenti il tasto premuto, il suo **scancode** e un eventuale evento di pressione ripetuta. Quindi, nel caso di **love.keypressed**, i valori **a**, **b** e **c** sarebbero definiti come qualcosa, mentre **d**, **e** ed **f** sarebbero nulli.

```
-- Aggiornare dt, poiché lo si passerà ad update
if love.timer then
  love.timer.step()
  dt = love.timer.getDelta()
end

-- Richiamare update e disegnare
if love.update then love.update(dt) end -- will pass 0 if
love.timer is disabled
```

love.timer.step misura il tempo tra gli ultimi due fotogrammi e cambia il valore restituito da **love.timer.getDelta**. In questo caso, quindi, **dt** conterrà il tempo impiegato per l'esecuzione dell'ultimo fotogramma. È utile perché questo valore viene passato alla funzione **love.update** e da lì può essere usato nel gioco per definire elementi con velocità costante, nonostante le variazioni della **frequenza dei fotogrammi**.

```
if love.graphics and love.graphics.isActive() then
  love.graphics.clear(love.graphics.
getBackgroundColor())
  love.graphics.origin()
  if love.draw then love.draw() end
  love.graphics.present()
end
```

Dopo **love.update** viene richiamato **love.draw**. Prima però si verifica che il modulo **love.graphics** esista e che si possa disegnare sullo schermo tramite **love.graphics.isActive**. Lo schermo viene riportato al colore di sfondo definito (inizialmente nero) con **love.graphics.clear**, le trasformazioni vengono reimpostate attraverso **love.graphics.origin**,

la funzione `love.draw` viene infine richiamata e poi `love.graphics.present` è usata per spingere sullo schermo tutto ciò che è stato disegnato in `love.draw`. Infine ci rimane:

```
if love.timer then love.timer.sleep(0.001) end
```

E con questo termina la funzione `love.run`. Tutto ciò che accade all'interno del **ciclo while true** viene considerato un fotogramma, il che significa che `love.update` e `love.draw` vengono richiamate una volta per ciascuno di essi. L'intero gioco si basa sulla ripetizione molto veloce del contenuto di questo ciclo (per esempio a **60** fotogrammi al secondo), quindi è importante comprenderlo bene.

Sfruttare le librerie disponibili

Prima di concludere l'articolo è importante parlare delle librerie **Lua/LÖVE** che possono servire per creare un videogioco. Una molto utile ed efficace è **rxi/classic** (**https://github.com/rxi/classic**), che si autodefinisce come "un piccolo modulo per le classi". Per installarla basta scaricarla e inserire la cartella **classic** nella directory del progetto. Può essere utile creare una cartella dedicata per inserire lì tutte le librerie. Una volta fatto questo, si può importare la libreria nel gioco all'inizio del file **main.lua** con:

```
Object = require 'libraries/classic/classic'
```

Come si legge nella sua pagina **GitHub**, con questa libreria si possono svolgere tutte le normali operazioni della **programmazione orientata agli oggetti** (**OOP**) e in genere funziona bene. Quando si crea una nuova classe, è generalmente bene farlo in un file separato e inserirlo in una cartella di oggetti. Per esempio, la creazione di una **classe Test** e la sua singola **istanziazione** si presentano in questo modo:

```
-- in objects/Test.lua
Test = Object:extend()

function Test:new()

end

function Test:update(dt)

end

function Test:draw()

end

-- in main.lua
Object = require 'libraries/classic/classic'
require 'objects/Test'

function love.load()
   test_instance = Test()
end
```

Quindi, quando la funzione `require 'objects/Test'` viene richiamata in **main.lua**, avviene tutto ciò che è definito nel file **Test.lua**, il che significa che la variabile globale `Test` contiene ora la definizione della classe `Test`. Per questo gioco, ogni definizione di classe è stata fatta in questo modo, il che significa che i nomi delle classi hanno dovuto essere unici, poiché sono legati a una variabile globale. Se volete evitare di procedere in questo modo, potete fare le seguenti modifiche:

```
-- in objects/Test.lua
local Test = Object:extend()
...
return Test

-- in main.lua
Test = require 'objects/Test'
```

Se definite la variabile `Test` come `locale` in **Test.lua**, non sarà legata a una variabile globale, il che significa che potete assegnarle il nome che volete quando la richiedete in main.lua. Alla fine dello script **Test.lua** viene restituita la variabile locale, quindi in **main.lua**, quando viene dichiarato `Test = require 'objects/Test'`, la definizione della classe `Test` viene assegnata alla variabile globale `Test`. A volte, per esempio se si scrivono librerie per terzi, questo è il modo migliore di procedere, per evitare di inquinare il loro stato globale con le variabili della propria libreria. Anche **Classic** fa così ed è per questo che bisogna inizializzarlo assegnandolo alla variabile `Object`. LXP

» IL LINGUAGGIO DELLA LUNA

Lua (**http://www.lua.org/**) è un linguaggio di **scripting** potente e leggero, **embeddabile** e che supporta la programmazione **procedurale**, **orientata agli oggetti** e **funzionale**. Il suo nome significa "Luna" in portoghese. Il linguaggio proviene infatti dal **Brasile** ed è progettato, implementato e aggiornato da un team della **PUC-Rio**, la **Pontificia Università Cattolica di Rio de Janeiro**. È spesso citato come il più veloce tra i linguaggi di scripting **interpretati** e può essere compilato senza bisogno di modifiche da parte dell'utente su tutte le piattaforme che dispongono di un **compilatore C** standard. Si può eseguire su tutte le versioni di **Unix** e **Linux** e su dispositivi mobili con **Android**, oltre che su vari sistemi operativi proprietari tra cui **Windows** e **iOS**. Un concetto fondamentale nella progettazione di Lua è quello di fornire **metameccanismi** per l'implementazione di funzioni, invece di integrarle. Per esempio, sebbene Lua non sia un linguaggio orientato agli oggetti puro, fornisce dei metameccanismi per implementare le **classi** e **l'ereditarietà**. Questo approccio porta a una riduzione dei concetti e a mantenere il linguaggio di dimensioni ridotte, consentendo al contempo di estenderne la semantica in modi non convenzionali. Va infine sottolineato che si tratta di un software Open Source, distribuito con **licenza MIT**.

Il logo di **Lua**, che significa "Luna" in portoghese, include il satellite

TIP

Per andare oltre al ciclo di gioco spiegato in queste pagine, all'indirizzo https://bit.ly/3hsJ8Ds potete trovare anche un eccellente articolo (in inglese) che esamina le diverse tecniche per i game loop.

L'eco dei LUG

I LUG

I LUG rappresentano da sempre il punto di riferimento per chiunque voglia conoscere GNU/Linux. Ogni mese dedicheremo loro questo spazio per la comunicazione di nuovi progetti e appuntamenti. Se hai qualcosa da segnalarci scrivi a **ecodeilug@linuxpro.it**

ABRUZZO

AnxaLUG - Lanciano
www.anxalug.org
Il Pinguino - Teramo
Non disponibile
OpenLUG - L'Aquila
Non disponibile
Pescara LUG
www.pescaralug.org
Pollinux LUG - Pollutri
Non disponibile
SSVLUG - San Salvo, Vasto, Termoli
www.ssvlug.org
TeateLUG - Chieti
Non disponibile
TeLUG - Teramo
www.telug.it
User Group Valle Roveto
http://linuxvalley-os4.blogspot.com/

BASILICATA

Basilicata LUG - Potenza e Matera
www.baslug.org
PLUG
www.pignolalug.it

CALABRIA

Bit01
www.associazionebit01.it
Bogomips - Bisignano
www.blug.it
CSLUG - Cosenza
http://cslug.linux.it
CzLug
Non disponibile
HackLab Catanzaro
http://hacklab.cz
HackLab Cosenza
https://hlcs.it
Piana LUG - Piana di Gioia Tauro
Non disponibile
SpixLug - Spezzano Albanese
Non disponibile
Verdebinario
www.verdebinario.org

CAMPANIA

GLUS
www.liberarete.it/glus
IGLUG - Napoli e provincia
www.iglug.org
IRLUG - Irpinia
www.irlug.it
LUG-Ischia
www.lug-ischia.org
LUG Acropoli
www.linux.it/~ciccios/lugagropoli.htm
Neapolis Hacklab
www.officina99.org/hacklab.html

EMILIA ROMAGNA

Borgotaro LUG - Val Taro
http://btlug.it/
ERLUG
http://erlug.linux.it
Ferrara LUG
www.ferrara.linux.it
FoLUG - Forlì
http://folug.linux.it
ImoLUG - Imola
www.imolug.org
PANLUG - Vignola
Non disponibile
PLUG - Parma
http://parma.linux.it
RELug - Reggio Emilia e provincia
http://relug.linux.it
RiminiLug
www.riminilug.it
S.P.R.I.Te
http://sprite.csr.unibo.it
UIELinux - Valle del Rubicone
www.uielinux.org

FRIULI VENEZIA GIULIA

GOLUG - Gorizia
www.golug.it
LUG Pordenone
www.pnlug.it
LugTrieste
http://trieste.linux.it

LAZIO

GioveLUG - Terracina
www.giovelug.org
Latina LUG
www.llg.it
LUG Privernum Volsca - Priverno (LT)
Non disponibile
LUG Rieti
https://rieti.ils.org
LUGRoma 3
www.lugroma3.org
TorLUG - Università Tor Vergata - Roma
http://lug.uniroma2.it/

LIGURIA

GE.P LUG - Genova
https://geplug.altervista.org/
Genova LUG
www.genovalug.altervista.org
Govonis GNU/LUG - Savona
www.govonis.org
TLug-TSL - Tigullio Ligure
http://tlug.linux.it/

LOMBARDIA

BGLug - Bergamo e provincia
www.bglug.it
BGLug Valle Seriana - Valle Seriana
http://bglugvs.web3king.com/
BrigX - Monza e Brianza
http://brigx.it
GL-Como - Como
www.gl-como.it
GLUX - Lecco e provincia
www.lecco.linux.it
GULLP - Gruppo Utenti Linux Lonate Pozzolo
www.gullp.it
HackLabCormano
http://hacklabcormano.it
LIFO - Varese
www.lifolab.org
LIFOS - Cinisello Balsamo
www.lifos.org
Linux Var - Varese
www.linuxvar.it
Lug8 - Gottolengo e Bassa Bresciana
http://lugotto.linux.it
LugBS - Brescia e provincia
http://lugbs.linux.it/
Lug Castegnato - Castegnato
www.kenparker.eu/LugCastegnato
LUG Legnano
https://luglegnano.wordpress.com
LugMan - Mantova e provincia
www.lugman.org

L'eco dei LUG

LugOB - Cologne e ovest bresciano
www.lugob.org
Lugotto - Gottolengo (BS)
http://lugotto.linux.it
POuL - Milano
www.poul.org
TiLug - Pavia
http://pavia.linux.it
VIMELUG - Linux User Group Vimercate
http://vimelug.org

MARCHE

CMlug
Non disponibile
FanoLUG
www.fanolug.org
GLM - Macerata
Non disponibile
PDP Free Software
https://pdp.linux.it

MOLISE

FrenterLUG - Larino
Non disponibile

PIEMONTE

BiLUG - Provincia di Biella
http://www.bilug.it
Gallug - Galliate
www.gallug.it
GLugTo
www.glugto.org
IvLug - Ivrea Linux User Group
www.ivlug.it
Linox Novara
www.linoxnovara.org

PUGLIA

BriLUG - Brindisi
Non disponibile
MurgiaLug - Santeramo in Colle
Non disponibile
ManfredoniaLug - Manfredonia
https://www.manfredonialug.it
SaLUG! - Salento
http://salug.it

SARDEGNA

GNUraghe - Oristano
www.gnuraghe.org
PLUGS - Sassari
Non disponibile

SICILIA

cLUG - Caltanissetta
Non disponibile
FreakNet MediaLab - Catania
www.freaknet.org
Free Circle
www.thefreecircle.org
Leonforte LUG
http://leonforte.linux.it
LUG Catania
www.catania.linux.it
LUGSR - Siracusa
www.siracusa.linux.it
MELUG - Messina
Non disponibile
Norp LUG - Noto, Pachino, Rosolini
Non disponibile
Poetry FreakNe
http://poetry.freaknet.org
VPLUG Linux Planet - Provincia Caltanisetta
www.vplug.it
SputniX - Palermo
www.sputnix.it

TOSCANA

ACROS - Area di Versilia, Lucca, Massa Carrara
www.lug-acros.org
Elbalinux
Non disponibile
ElsaGLUG - Val d'Elsa
www.elsaglug.org
FLUG - Firenze
www.firenze.linux.it
GOLEM - Empoli, Valdelsa
http://golem.linux.it
G.U.L.LI - Livorno
www.livorno.linux.it
GULP Pisa
www.gulp.linux.it
GuruAtWork - Grosseto e provincia
www.guruatwork.com
Lucca LUG
http://luccalug.it
L.U.G.A.R - Arezzo
Non disponibile
PtLug - Pistoia e provincia
www.ptlug.org
SLUG - Siena e provincia
www.siena.linux.it

TRENTINO ALTO ADIGE

LinuxTrent - Trento
http://linuxtrent.it
LugBz - Bolzano
www.lugbz.org

UMBRIA

OrvietoLUG
www.orvietolug.it
LUG Perugia
www.perugiagnulug.org

VALLE D'AOSTA

SLAG - Aosta
Non disponibile

VENETO

0421ug - Provincia di Venezia
www.0421ug.org
BLUG - Belluno
http://belluno.linux.it
GrappaLUG - Bassano del Grappa
https://grappalug.org/
LegnagoLUG
Non disponibile
Linux Ludus - Villafranca (VR)
www.linuxludus.it
LugAnegA
www.luganega.org
MontelLUG - Montebelluna
www.montellug.it
FSUG Padova
www.fsugpadova.org
TVLUG - Treviso
www.tvlug.it
VELug - Venezia
www.velug.it
AViLUG Schio
Non disponibile

NAZIONALI

Gentoo Channel Italia
www.gechi.it

In edicola dal 10 febbraio

NEL PROSSIMO NUMERO

Hai un argomento da proporci?
Scrivi a **redazione@linuxpro.it**

Scegli la lingua giusta

Ecco come selezionare il linguaggio di programmazione giusto per ogni tipo di esigenza

Bimestrale - prezzo di copertina 6,90 €
www.linuxpro.it- redazione@linuxpro.it

La Divisione Informatica di Sprea edita anche:
WIN MAGAZINE + UBUNTU FACILE
MAC IDEA! + APP JOURNAL + HACKER JOURNAL

Business Unit Manager: Massimiliano Zagaglia

Cover: Luca Patrian

Realizzazione editoriale a cura di: **Dehive Sas di Andrea Orchesi**

Sprea S.p.A.
Sede Legale: Via Torino, 51 20063 Cernusco Sul Naviglio (Mi) - Italia
PI 12770820152- Iscrizione camera Commercio 00746350149
Per informazioni, potete contattarci allo 02 924321

CDA:
Luca Sprea (Presidente), Alessandro Agnoli (Amministratore Delegato), Giulia Spreafico (Divisione digital), Stefano Pernarella (ADV & PR)

ADVERTISING, SPECIAL PROJECTS & EVENTS
Segreteria: Emanuela Mapelli - Tel. 02 92432244 - emanuelamapelli@sprea.it

SERVIZIO QUALITÀ EDICOLANTI E DL
Sonia Lancellotti, Luca Majocchi : Tel. 02 92432295
distribuzione@sprea.it 351 5582739

ABBONAMENTI E ARRETRATI
Abbonamenti: si sottoscrivono on-line su www.sprea.it/linuxpro
abbonamenti@sprea.it
Tel 02 87168197 (lun-ven / 9:00-13:00 e 14:00-18:00)
Il prezzo dell'abbonamento è calcolato in modo etico perché sia un servizio utile e non in concorrenza sleale con la distribuzione in edicola.
Arretrati: si acquistano on-line su www.sprea.it/arretrati
abbonamenti@sprea.it Tel 02 87168197 (lun-ven / 9:00-13:00 e 14:00-18:00)
329 3922420

FOREIGN RIGHTS
Paolo Cionti: Tel. 02 92432253 - paolocionti@sprea.it

SERVIZI CENTRALIZZATI
Art director: Silvia Taietti
Grafici: Alessandro Bisquola, Nicole Bombelli, Tamara Bombelli, Nicolò Digiuni, Marcella Gavinelli, Luca Patrian
Coordinamento: Chiara Civilla, Tiziana Rosato, Roberta Tempesta, Silvia Vitali
Amministrazione: Erika Colombo (responsabile), Silvia Biolcati, Irene Citino, Desirée Conti, Sara Palestra- amministrazione@sprea.it
Ufficio Legale: Francesca Sigismondi

Linux Pro, pubblicazione registrata al Tribunale di Milano il 08.02.2003 con il numero 74. ISSN: 1722-6163

Direttore responsabile: Luca Sprea

Distributore per l'Italia: Press-Di Distribuzione stampa e multimedia s.r.l. 20090 Segrate

Distributore per l'Estero: SO.DI.P S.p.A. Via Bettola, 18 - 20092 Cinisello Balsamo (MI)
Tel. +390266030400 - Fax +390266030269 - sies@sodip.it - www.sodip.it

Stampa: Arti Grafiche Boccia S.p.A. Via Tiberio Claudio Felice, 7- 84131 Salerno